Gould Computer Systems

Diagnostic Manual

MP MASTER
MA34645

GOULD
Electronics

Gould CONCEPT/32™

Diagnostic Facility

Documentation Set

Book 4 of 5

Publication Order Number: 326-005450-000

™CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

Gould CONCEPT/32™

Diagnostic Executive Program

(CN.DXP.r)

Diagnostic Description

Diagnostic Number: 111032-900r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005670-000

™ CONCEPT/32 is a trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Diagnostic Executive Program (CN.DXP.r) Diagnostic Description Release 1.0, Publication Order Number 326-005670-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vii/viii
1-1 through 1-4
2-1 through 2-8
3-1 through 3-17/3-18
4-1 through 4-38
5-1 through 5-12

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_06203

CONTENTS

CHAPTER 1 SYSTEM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-2
1.3 Software Components 1-2
1.4 Memory Utilization 1-3
1.5 Executive Services 1-3
1.6 Automatic System Elapsed Time Processor 1-3
1.7 File Input/Output Operations 1-4
1.7.1 File Access 1-4
1.7.2 File Identification 1-4
1.7.3 System File Code Assignments 1-4

CHAPTER 2 LOADING AND EXECUTION

2.1 Loading DEXP 2-1
2.2 DEXP Initialization 2-2
2.3 Cold Start Procedure 2-3
2.4 Warm Start Procedure 2-3
2.5 Loading Programs Under DEXP 2-4
2.6 The Load Map 2-5
2.7 The Load Task 2-6
2.8 Using Options 2-6
2.8.1 Timesliced Program Options 2-7
2.8.2 DEXP Option Set 2-8
2.9 Terminating Execution 2-8

CHAPTER 3 MESSAGE, INTERRUPT, AND TRAP PROCESSING

3.1 Milestone Messages 3-1
3.2 Error Messages 3-3
3.3 Load Task Error Messages 3-6
3.4 Interrupt and Trap Processing 3-8
3.5 System Integrity Traps and Major System Interrupts 3-9
3.5.1 Power Fail-Safe Trap 3-10
3.5.2 Power Auto-Start Trap 3-10
3.5.3 Memory Parity Error Trap 3-10
3.5.4 Nonpresent Memory Error Trap 3-10
3.5.5 Undefined Instruction Error Trap 3-10
3.5.6 Privileged State Violation Trap 3-10
3.5.7 Supervisor Call Trap 3-11
3.5.8 Machine Check Trap 3-11
3.5.9 System Check Trap 3-11
3.5.10 Map Fault Trap 3-11
3.5.11 I/O Instruction Trap 3-11

3.5.12 SIPU Instruction Trap 3-11
3.5.13 Address Specification Trap 3-11
3.5.14 Attention Interrupt 3-12
3.5.15 Privileged Halt Trap 3-12
3.5.16 Arithmetic Exception Trap 3-12
3.5.17 Cache Error Trap 3-12
3.5.18 Real-Time Clock Interrupt 3-12
3.5.19 Spurious Interrupt Receiver 3-12
3.6 Executive Service (SVC) Abort Codes 3-13
3.7 Run-Time Abort Codes 3-14
3.8 I/O Handler Abort Codes 3-15
3.9 Compile-Time Abort Codes 3-16
3.10 Programmed Halts 3-17

CHAPTER 4 DEXP COMMANDS

4.1 Command Overview 4-1
4.2 Command Arguments 4-3
4.3 Command Syntax 4-5
4.4 Editing Commands 4-5
4.5 Type-Ahead Function 4-6
4.6 ABORT Command 4-7
4.7 ASSIGN Command 4-8
4.7.1 File Assignment Derivatives 4-8
4.7.2 Program Assignment Derivatives 4-9
4.8 BATCH Command 4-10
4.9 BIAS Command 4-11
4.10 CONTINUE Command 4-11
4.11 DEFINE Command 4-12
4.12 DELETE Command 4-13
4.13 DUMP Command 4-14
4.13.1 Executive Dump Task (S.DUMP) 4-14
4.13.2 Dump Output 4-14
4.13.3 Zero Relative Dump Derivative 4-15
4.13.4 Task Relative Dump Derivative 4-15
4.13.5 PSDS, REGS, and TCA Dump Derivatives 4-16
4.13.6 Scratchpad Dump Derivative 4-16
4.13.7 Postmortem Dump Controls 4-17
4.13.8 Directory Derivative 4-17
4.13.9 FDT, DCB, and IOEA Dump Derivatives 4-17
4.14 EJECT Command 4-18
4.15 EXAMINE Command 4-18
4.16 EXIT Command 4-19
4.17 GOTO Command 4-19
4.18 HELP Command 4-20
4.19 HOLD Command 4-20
4.20 LOAD Command 4-21
4.21 MODIFY Command 4-22
4.22 MODIFY SPAD Command 4-22
4.23 NOTE Command 4-23
4.24 PAGESIZE Command 4-23
4.25 REINITIALIZE Command 4-24

4.26 RESET Command 4-25
4.26.1 Option Word Derivative 4-25
4.26.2 Bit Manipulation Derivative 4-25
4.27 RESTART Command 4-26
4.28 REWIND Command 4-28
4.29 SEARCH Command 4-28
4.30 SET Command 4-29
4.30.1 Option Word Derivative 4-29
4.30.2 Bit Manipulation Derivative 4-29
4.31 SHOW Command 4-30
4.31.1 Macro Derivative 4-30
4.31.2 Options Derivative 4-30
4.31.3 Show Directory Derivative 4-30
4.31.4 Test Derivative 4-31
4.31.5 Show File Derivative 4-31
4.32 SNAP Command 4-32
4.33 START Command 4-33
4.34 STATUS Command 4-35
4.34.1 Automatic Status Derivative 4-35
4.34.2 Spurious Interrupt Derivative 4-35
4.34.3 Task Status Derivative 4-36
4.35 TEST Command 4-37
4.35.1 Option Word Derivative 4-37
4.35.2 Bit Manipulation Derivative 4-37
4.36 TIME Command 4-38
4.37 UPSPACE Command 4-38
4.38 WAIT Command 4-38

CHAPTER 5 DIRECTIVE LANGUAGE

5.1 Directive Language Overview 5-1
5.2 Creating Directive Tests 5-1
5.3 Editing Directive Tests 5-1
5.4 Using Directives 5-2
5.5 Using Arguments 5-3
5.6 Directive Macros 5-3
5.7 Basic Directive Types 5-3
5.7.1 Simple Directives 5-3
5.7.2 Literal Directives 5-4
5.7.3 Parameter Directives 5-4
5.7.4 Jump Directives 5-5
5.8 Standard Directives 5-5
5.8.1 "A" Directives (Assign) 5-6
5.8.2 "C" Directives (Cycle) 5-6
5.8.3 "D" Directives (Dump) 5-7
5.8.4 "H" Directives (Hold) 5-7
5.8.5 "J" Directives (Jump) 5-8
5.8.6 "L" Directives (Load) 5-9
5.8.7 "P" Directives (Parameter) 5-11
5.8.8 "S" Directive (Stop) 5-12
5.8.9 "T" Directive (Time) 5-12

CHAPTER 1

SYSTEM OVERVIEW

1.1 General Overview

The Gould CONCEPT/32 Diagnostic Executive Program (DEXP) is a diagnostic operating system that supports concurrent execution of multiple level two macrodiagnostic programs in an interactive environment.

The DEXP's directive language compiler and directives supplied by level two macrodiagnostic programs provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The DEXP includes the following system features:

- Minimum memory usage (128KW)
- Reentrant services
- Modularity (the DEXP and programs share all executive services)
- Common syntax for command and directive language processing
- Minimum hardware requirements
- Uniform program structure and DEXP interface
- Support for all standard product devices and options
- Full 16-megabyte addressability
- Standard handling for all interrupts and traps
- Interprogram communication through general purpose registers and memory partitions

1.2 Resource Requirements

The Gould CONCEPT/32 Diagnostic Executive Program (DEXP) requires the following hardware components:

- A CONCEPT/32 CPU with a minimum of 64K words of memory
- A CONCEPT/32 IPU (optional)
- An IOP

1.3 Software Components

The Gould CONCEPT/32 Diagnostic Executive Program (DEXP) includes the following software components:

. Diagnostic Executive Task	(CN.DXP.r)
. Operator Communications Task	(S.OPCOM)
. Load Task	(S.LOAD)
. Dump Task	(S.DUMP)
. Teletypewriter (CRT) Handler	(H.TY)
. Eight-Line Handler	(H.UT)
. Line Printer Handler	(H.LP)
. Card Reader Handler	(H.CR)
. Magnetic Tape Handler	(H.MT)
. Disc Handler	(H.DC)
. Remote Testing Handler	(H.CA)
. Interrupt and Trap Handlers	
Power-Fail Trap Handler (RPL00)	(CPU:PWRD)(IPU:PWRD)
Auto-start Trap Handler (RPL01)	(CPU:PWRU)(IPU:PWRU)
Memory Parity Trap Handler (RPL02)	(CPU:MPAR)(IPU:MPAR)
Nonpresent Memory Trap Handler (RPL03)	(CPU:NMEM)(IPU:NMEM)
Undefined Instruction Trap Handler (RPL04)	(CPU:UNDF)(IPU:UNDF)
Privilege Violation Trap Handler (RPL05)	(CPU:PRIV)(IPU:PRIV)
Supervisor Call Trap Handler (RPL06)	(CPU:SVCT)(IPU:SVCT)
Machine Check Trap Handler (RPL07)	(CPU:MCHK)(IPU:MCHK)
System Check Trap Handler (RPL08)	(CPU:SCHK)(IPU:SCHK)
Map Fault Trap Handler (RPL09)	(CPU:MFLT)(IPU:MFLT)
Address Specification Trap Handler (RPL0C)	(CPU:ADDR)(IPU:ADDR)
Attention Trap Handler (RPL0D)	(CPU:ATTN)(IPU:ATTN)
Privilege Halt Trap Handler (RPL0E)	(CPU:PRIH)(IPU:PRIH)
Arithmetic Exception Trap Handler(RPL0F)	(CPU:ARIT)(IPU:ARIT)
Cache Error Trap Handler (RPL10)	(CPU:CACH) (IPU:CACH)
Real-Time Clock Interrupt Handler (PL18)	(RTCLOCK)

1.4 Memory Utilization

The basic unit of memory allocation to a program is 512 words. As the program loads, the DEXP allocates memory to it. The DEXP makes no provisions for dynamic memory allocation, although some programs solicit memory interactively during their initialization process.

The DEXP supports the full 16-megabyte addressability of the CONCEPT/32 hardware architecture. However, it supports only one 128K word of execution space. The DEXP runs unmapped to reduce system complexity during troubleshooting.

Since level two microdiagnostic programs have their own I/O instructions, they execute in the privileged mode. Compute-bound programs execute in either the privileged or unprivileged mode.

If performing memory management functions (Examine, Modify, Dump, etc.), the DEXP ensures that the addresses specified are within physical memory. This allows maximum flexibility if systems are configured with shared and/or discontiguous memory. However, if a memory management function is performed in nonpresent physical memory, a nonpresent memory trap almost results as the DEXP attempts to perform all operations as instructed.

1.5 Executive Services

The DEXP provides executive service routines as standard modular components to perform frequently required operations with maximum efficiency. The DEXP and programs share executive services with most executive service routines available to an active program.

1.6 Automatic System Elapsed Time Processor

If DEXP Option 2 (enables status data display) is set or the Automatic Status Processor is ON (which automatically sets DEXP Options 1 and 2), the automatic system elapsed time processor writes the system elapsed time to the listed output (LO) file every ten minutes.

By writing the time every ten minutes, the system keeps an accurate log (+ or -10 minutes) of the duration of program testing. To obtain the system elapsed time at any time, use the DEXP Time command.

If the LO file is assigned to the same device as the operator communications (OC) file, the automatic elapsed time processor does not write its message.

In addition, the automatic system elapsed time processor drives the automatic task status enquiry processor. For more information, refer to the DEXP Status command.

1.7 File Input/Output Operations

The DEXP provides the basic I/O services and management functions such as writing error messages, writing milestone messages, and issuing prompts for required parameters prior to program execution. While keeping program overhead to a minimum, the DEXP file system receives and processes all I/O requests, performs logical error checking and parameter validation, and assigns the appropriate device handler to the I/O request.

The individual device handlers execute data exchanges, process service interrupts, and perform device testing.

1.7.1 File Access

The DEXP implements all I/O operations as Wait I/O operations. However, the system is not idle during I/O operations, only the program requesting the I/O operation is idle.

1.7.2 File Identification

System file codes contain two ASCII characters. For each file code there exists a file definition table (FDT). The FDT entries define the physical device assigned to the file, the channel priority level, channel address, device subaddress, file status, and information pertaining to the reserved or free state of the file. Absolute memory locations 800_{16} through $8DC_{16}$ store the FDTs.

1.7.3 System File Code Assignments

All file I/O operations in the system are to and from system specified file codes. The program assigns and reassigns these system file codes to the physical devices where the DEXP routes the I/O commands through the Assign command.

The following are the system file code assignments for the DEXP:

File Code		Default Assignment	Reassignments
BI	Binary Input File	IPL device (for loading tasks)	
BO	Binary Output File	null device	
CC	Command Control File	IPL device	
LO	Listed Output File	OC File	any output device
OC	Operator Communications File	operator's console	teletypewriter user terminal remote communications adapter
OL	Overlay Loader File	IPL device	can not be reassigned
SI	Scratch Input File	IPL device	
SO	Scratch Output File	null device	

CHAPTER 2

LOADING AND EXECUTION

2.1 Loading DEXP

The Gould CONCEPT/32 Diagnostic Executive Program (DEXP) loads similarly to a level one macrodiagnostic program from the Diagnostic Overlay Loader (DOL) by performing the following two-step operation:

1. Perform an Initial Program Load (IPL) procedure to load the Diagnostic Overlay Loader.

   The Diagnostic Overlay Loader then writes the following start message and a DOL> prompt to the operator communications (OC) file:

   ```
   GOULD CSD DIAGNOSTIC LOADER (DOL #001030-102r date)
   (C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
   ALL RIGHTS RESERVED

   DOL >
   ```

2. Enter the following:

   ```
   CN.DXP.r
   ```

   r specifies the current program revision level.

   The Diagnostic Executive Program writes the following start message and its initialization sequence prompts:

   ```
   GOULD CSD DIAGNOSTIC EXECUTIVE (DXP #111032-900r date)
   (C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
   ALL RIGHTS RESERVED
   ```

2.2 DEXP Initialization

As the DEXP begins execution, it writes the following initialization sequence prompts to the operator communications (OC) file:

Note: The DEXP reissues any incorrectly answered initialization sequence prompts.

1. The DEXP issues the following prompt:

   DEFAULT REAL-TIME CLOCK RATE = 60 HZ; [CR, ##]?

   Responding with the Return key causes the program to assign the specified default real-time clock rate (60). To modify the real-time clock rate, enter a valid clock rate (50, 100, 120).

2. The DEXP then issues the following prompt:

   DEFAULT ELAPSED-TIME = 00:00:00; [CR, HH:MM:SS]?

   Responding with the Return key causes the program to assign the specified default system elapsed time (00:00:00). To modify the default system elapsed time, enter a valid time in hours (HH), minutes (MM), and seconds (SS).

   If the time-of-day or any other value is entered, the times indicated during execution represent the system elapsed time plus the initial time setting.

3. The DEXP then issues the following prompt:

   DO YOU NEED HELP; [Y, N]?

   Responding with Y (yes) causes the program to write a help message, a list of DEXP commands, and the DXP> prompt to the operator communications (OC) file. Responding with N (no) or pressing the Return key causes the program to issue the DXP> prompt.

2.3 Cold Start Procedure

Main memory location 0_{16} contains the cold start entry point address. The system performs the cold start procedure automatically as part of the initial program load (IPL) sequence. Perform a cold start at any time from the console using the following steps:

1. Check for the Panel mode prompt (//) and if the prompt is not the last line displayed, enter ^aP.
2. Enter HALT to halt the system.
3. Enter RST to reset the system.
4. Enter PC=8.
5. Enter RUN to start the system.

Following a cold start, the DEXP performs as if it were initially loaded by writing its start message and initialization prompts to the operator communications (OC) file. No level two macrodiagnostic programs reside in memory.

If no level two macrodiagnostic programs are active in the system, perform a system cold start by issuing the Reinitialize command specifying CN.DXP.r as the taskname. If level two macrodiagnostic programs are active and an attempt is made to Reinitialize CN.DXP.r, the program writes the following message to the OC file:

```
*DIAGNOSTIC ACTIVE:  COMMAND IGNORED
```

2.4 Warm Start Procedure

Main memory location 0_{16} contains the warm start entry point address. Perform a warm start any time from the console by following the cold start procedure except set the program counter to zero (PC=0).

The DEXP writes its start message, followed by the default system elapsed time prompt, but does not issue the Real-Time Clock rate prompt and does not perform memory sizing (the creation of a memory module map).

During a warm start, all tasks become inactive. The DEXP reestablishes all interrupt and trap entry points and enables their priority levels. The DEXP also sets the CPU mode to reflect the version of the CONCEPT/32 system being tested, reads the system scratchpad into the CPU, and purges all system files.

If no level two macrodiagnostic programs are active in the system, perform a system warm start by issuing the Restart command specifying CN.DXP.r as the taskname. If level two macrodiagnostic programs are active and an attempt is made to Restart CN.DXP.r, the program writes the following message to the operator communications (OC) file:

```
* DIAGNOSTIC ACTIVE: COMMAND IGNORED
```

2.5 Loading Programs Under DEXP

The DEXP allows simultaneous execution of any number of level two macrodiagnostic programs (limited to 512K bytes of memory) while providing direct operator control for fault isolation and troubleshooting.

Loading Programs from the IPL Device

To load a program from the Diagnostic Object File (DOF) created by the automatic generation program (AUTOGN.r), enter the following in response to the DXP> prompt:

DXP>LOAD [taskname][(filename)][,address][@typa$_1$;[...;typa$_n$;]]

taskname	specifies a one to eight character name given to the task.
(filename)	specifies the eight-character name corresponding to the Diagnostic Facility Directory entry. The command syntax requires that the argument be enclosed in parentheses.
address	specifies a load address if the task is to be loaded at other than the default program counter address.
@	indicates that the data to follow is type-ahead data.
typa	specifies the type-ahead data. Delimit individual type-ahead entries with a semicolon.

Loading Programs from other than the IPL Device

To load a program from other than the IPL device, reassign the binary input (BI) file (its default assignment is to the IPL device) using the Assign command.

If loading from magnetic tape or disc, the files's coordinates must be known and entered during the load process.

If the binary input (BI) file is assigned to a magnetic tape, the absolute file coordinate corresponds to the number of files (in decimal) to be skipped prior to the desired file being positioned correctly. For example, if the second file on the tape is to be loaded, enter a 1.

If the binary input (BI) file is assigned to a disc, enter the absolute cylinder, head, and sector address coordinates. For example, to load the task residing at cylinder 36, head 5, and sector 3, enter 360503.

Mount the disc or tape containing the file and load the file using the DEXP Load command. The DEXP then writes the following message:

* WARNING: FILE NOT FOUND IN DIRECTORY

During the load process, the DEXP issues the following prompt to the operator communications (OC) file:

```
'handler 'MOUNT TASK' progname' ON ttaa-dsa; [C,H,T,#,U] ?
```

progname	specifies the eight-character name corresponding to the directory entry. (If the file is not on the IPL tape, the DEXP writes "SCRATCH" instead of the filename.)
ttaa	specifies the four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.
dsa	specifies the two-character hexadecimal device subaddress.

If loading from the IPL device, respond with C to continue execution, with H to continue execution, or with T to terminate execution. If loading from other than the IPL device, respond with the relative file position, the ordinates of the file (#), or with U (specifying an unblocked file). The DEXP loads the file and continues execution.

2.6 The Load Map

After loading a level two memory-resident program, the DEXP writes the following load map to the listed output (LO) file:

```
*LOADED : 'taskname'
 PGM : address, length
```

taskname	specifies the one to eight character name indicated in the Load command.
PGM	indicates a program.
address	specifies the hexadecimal address where the program is loaded.
length	specifies the program length (in decimal bytes).

If input and output buffers are assigned, the DEXP writes the following messages:

```
BFI: address , length
BFO: address , length
```

BFI	indicates the task's input buffer segment.
BFO	indicates the task's output buffer segment.
address	specifies the hexadecimal address of the indicated buffer segment.
length	specifies the length of the indicated buffer segment (in decimal bytes).

After writing the load map, the DEXP writes the following message:

```
*WARNING:  TASK 'progname' NOW 'taskname'
```

All subsequent references to the program must specify 'taskname'.

2.7 The Load Task

The DEXP then processes the following loader function codes (LFC) in order:

LFC	Function
0	Absolute data
1	Program origin
3	Transfer address
4	Relocatable data
5	Program name and boundary requirements
F	Last object byte

During processing of loader function code type 3 (transfer address), the DEXP transfers control to the program just loaded at the transfer address. If no transfer address is specified, the load task terminates.

The transfer address for a program is usually the initialization entry point. Therefore, program initialization occurs at program load time.

After completing the initialization process, the loader immediately terminates and the task is available for execution.

At this time, set options and/or use the DEXP Start command to begin level two macrodiagnostic execution.

2.8 Using Options

The DEXP and level two macrodiagnostic programs contain options which modify execution conditions depending upon whether they are set or reset. To set, reset, or test the Diagnostic Executive Program (DEXP) options and macrodiagnostic program options, use the following commands:

To reset program options, enter the following:

DXP>RESET taskname OPT, opt_1 [,...,opt_n]

To set program options, enter the following:

DXP>SET taskname OPT, opt_1 [,...,opt_n]

To test program options, enter the following:

DXP>TEST taskname OPT, opt_1 [,...,opt_n]

OPT indicates that the function specified is to be performed on the option word.

opt specifies the particular options to be set, reset, or tested.

For the test function only, the DEXP writes the following message to the listed output (LO) file and repeats it for each specified option:

OPT##: {SET / RES}

If the DEXP filename (CN.DXP.r) or no taskname is given, the DEXP performs the Set, Reset, and Test commands on its own options.

2.8.1 Standard Program Options

The following options are standard to most level two macroinstruction programs. Use the DEXP Set and Reset commands to change default settings, thereby modifying execution conditions. Refer to the individual diagnostic descriptions for additional options specific to that program.

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence detecting an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Inhibits operator intervention tests. Setting this option inhibits any test requiring operator intervention.
9	Reset	Reserved, do not use.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file.
11	Set	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Inhibits error message expansion.

2.8.2 DEXP Option Set

The DEXP contains the following options:

Option	Default	Effect If Set
0	Reset	Enables post-mortem dumps
1	Reset	Enables status data displays every 10 minutes
2	Reset	Enables system performance data displays
3	Reset	Inhibits system mount messages
4	Reset	Inhibits expanded help messages
5	Reset	Inhibits batch mode break messages
6	Reset	Enables remote automatic testing

Note: Setting Options 1 and 2 is similar to setting the DEXP status command ON for automatic status displays every 10 minutes. If the status command is ON, the program sets Options 1 and 2 automatically.

2.9 Terminating Execution

To terminate program execution, use one of the following methods:

DEXP Abort command

```
DXP>ABORT taskname
```

The DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
DXP>HOLD taskname
```

The DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes the DEXP to execute the commands and issue a DXP> prompt.

To terminate the DEXP, use the Exit command.

DEXP Exit command

```
@@A
DXP>EXIT
```

If all programs are inactive, the DEXP terminates and returns control to the Diagnostic Overlay Loader (DOL).

CHAPTER 3

MESSAGE, INTERRUPT, AND TRAP PROCESSING

3.1 Milestone Messages

If a program has Option 11 (writes milestone messages) set, the DEXP writes the following messages:

If a Start command is issued, the DEXP writes the following message:

```
hh:mm:ss 'taskname' STARTED
```

If a Hold command is issued, the DEXP writes the following message:

```
hh:mm:ss 'taskname' HELD
```

To resume execution, use the DEXP Continue command.

If a Continue command is issued, the DEXP writes the following message:

```
hh:mm:ss 'taskname' CONTINUED
```

If an Abort command is issued, the DEXP writes the following message:

```
hh:mm:ss 'taskname' ABORTED
```

After starting each program test, the DEXP writes the following message:

```
hh:mm:ss 'taskname' TEST: (#) optional test definition
```

After completing each program pass, the DEXP writes the following message:

```
hh:mm:ss 'taskname' FINISHED PASS: (#)
```

If Option 6 (repeats program) is reset and a program pass is completed, the DEXP writes the following message:

```
'taskname' WHAT NEXT??
```

The DEXP also writes the previous message after completing directive string execution.

If a program requests an executive service suspension with no specified time interval, the DEXP writes the following message:

```
'taskname' message; [C,H,T]?
```

Respond with C to continue execution or use the DEXP Continue command. Respond with H to continue suspension, or with T to terminate execution.

If a task opens a system file, the DEXP writes the following task separation banner to the listed output (LO) file:

```
<<<<' taskname'----hh:mm:ss>>>>
```

If a Test command is issued, the DEXP writes the following message to the operator communications (OC) file:

```
{OPT}  ##: {SET}
{BIT}      {RES}
```

The DEXP repeats this message for each specified option or bit.

If a Examine, Snap, or Search command is issued, the DEXP writes the following message (and repeats it for each specified address):

```
pppppp (Rrrrrr)vvvvvvvv
```

pppppp specifies the hexadecimal physical address of the data.

Rrrrrr specifies the hexadecimal address of the data (relative to the program).

vvvvvvvv specifies the hexadecimal contents of the memory location.

If a Stop directive is issued, the DEXP writes the following message:

```
'taskname',STPn, DIRECTIVE...; [C,H,T] ?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

The number (n) displayed in the message signifies which Stop directive within the directive string causes the suspension.

3.2 Error Messages

If an error condition occurs, the DEXP writes one of the following messages:

If an attempt is made to load or reinitialize the Diagnostic Executive Program (DEXP) while a program is active in the system or if the Time command is issued while a level two macrodiagnostic program executing, the DEXP writes the following message:

```
*DIAGNOSTIC ACTIVE: COMMAND IGNORED
```

If an IPU is specified but is not configured in the system, the DEXP writes the following message:

```
*IPU NOT PRESENT
```

If the task specified is not in the system, the DEXP writes the following message:

```
*TASK NOT FOUND
```

If the task specified is not a level two macrodiagnostic program, the DEXP writes the following message:

```
*TASK NOT A DIAGNOSTIC
```

If a command for an active task is issued to an inactive task, the DEXP writes the following message:

```
*TASK NOT ACTIVE
```

If a command for an inactive task is issued to an active task, the DEXP writes the following message:

```
'taskname' ACTIVE ; [C,T]?
```

Responding with C continues processing the command and terminates the task, T terminates the command and continues the task.

If an invalid command verb or taskname is issued, the DEXP writes the following message:

```
*COMMAND NOT RECOGNIZED
```

If the physical device assigned to a system file is inoperable, the DEXP writes the following message:

```
'H.tt 'ttaa-dsa DEVICE INOPERABLE
```

If the physical device assigned to a system file reports any unusual condition other than device inoperable, the DEXP writes the following message:

```
'H.tt 'TERMINATED:ttaa-dsaSTATUS********_********ssssssssssssssss
```

If an invalid argument is specified with a command, the DEXP writes the following message:

*ARGUMENT kw(sf) INVALID

kw(sf) specifies the position (in decimal) of the invalid argument within the command string. This value specifies the positions of both the keyword and subfield portions of the invalid argument.

The following conditions cause arguments to be invalid:

- Argument out of range
- Argument not a file code (where file code is required)
- Argument not hexadecimal (where hexadecimal number is required)
- Argument not decimal (where decimal number is required)
- Argument not numeric (where numeric is required)
- Incorrect argument specification
- Illegal file assignment
- Illegal channel assignment
- Illegal device subaddress assignment
- Attempt to activate a system task
- Attempt to activate a device handler
- Device handler not present
- Test number not decimal
- Test number not defined
- No test number found
- No test numbers allowed
- No inclusive separator found
- Invalid tag present
- First address greater than last address

If an illegal file code is specified or if an illegal count is specified (count not decimal), the DEXP writes the following message:

*ARGUMENT kw(sf) NOT RECOGNIZED

If a required command argument is not specified, the DEXP writes the following message:

*ARGUMENT kw(sf) MISSING

If an attempt is made to reassign the operator communications (OC) file to a file code other than UT, TY, or CA, the DEXP writes the following message:

```
•CAN'T REASSIGN 'OC' FILE TO REQUESTED DEVICE
```

If the command control (CC) file is assigned to a null device and a Batch command is issued, or if the source input (SI) file is assigned to a null device and a Show command is issued, the DEXP writes the following message:

```
•IMPROPER 'filecode' FILE ASSIGNMENT, REASSIGN
```

If an attempt is made to assign a file while the file is in use, the DEXP writes the following message:

```
•FILE IN USE:ASSIGN IGNORED
```

If directive processing errors are detected, the DEXP writes one of the following messages to the operator communications (OC) file and terminates the program:

If a directive compilation or run-time error occurs, the DEXP writes the following message:

```
'taskname' TERMINATED: code DIRTV: {directive in error}
                                   {NONE              }
```

NONE indicates that a directive test is not being executed.

If an error occurs during directive compile time, the DEXP writes the following message:

```
'taskname' TERMINATED:  CTxx DIRTV:  directive
```

taskname specifies the one to eight character name of the program.

CTxx indicates a compile time termination error code (refer to the Section on Directive Compile Time Abort Codes).

directive specifies the directive being compiled at the time the error is detected.

If the error occurs during directive run-time, the DEXP writes the following message:

```
'taskname' TERMINATED:  RTxx DIRTV: directive
```

taskname specifies the one to eight character name of the program.

RTxx indicates a run-time termination error code (refer to the Section on Directive Run-time Abort Codes).

directive specifies the directive being executed at the time the error is detected.

3.3 Load Task Error Messages

If an error occurs during the DEXP Executive Load Task (S.LOAD) execution, the DEXP writes one of the following messages:

If a task being loaded reaches the 512 Kilobyte (128 Kiloword) execution space boundary, the DEXP writes the following message:

```
'S.LOAD' ' TERMINATED:  512KB EXECUTION SPACE EXCEEDED
```

The previous message indicates there is insufficient space left in main memory to load the task.

If an illegal loader function code is specified in the last object record read by the loader, the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  ILLEGAL LFC(#)
```

If the record control byte (the first byte of each object record) is other than a binary record (DF_{16}) or a last binary record (FF_{16}), the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  ILLEGAL RECORD CONTROL BYTE (##)
```

If the task being loaded is not formatted for the Program Status Doubleword (PSD) mode of operation, the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  TASK 'taskname' NOT PSD MODE FORMATTED
```

If a program name or the taskname is not found in the binary input (BI) file, the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  TASK 'taskname' NOT FOUND
```

If the address specified during a Load command causes a program overlay, the DEXP writes the following message:

```
'S.LOAD 'TERMINATED:  ILLEGAL ORIGIN(address)
```

If the binary input (BI) file is assigned to a null device, the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  "BI" FILE NOT ASSIGNED
```

If an attempt is made to load a level one macrodiagnostic program under DEXP instead of DOL, the DEXP writes the following message:

```
'S.LOAD ' TERMINATED:  TASK 'progname' NOT LEVEL 2
```

If an attempt is made to load a task under the same name as an already resident task, the DEXP writes the following message:

```
'S.LOAD' TASK 'progname' IN USE ; [C,H,T]?
```

Responding with C continues processing the command, and aborts the task. Responding with H places the task on hold, T terminates the load process.

If the check sum calculated by the loader does not equal the check sum in the last object record read, the DEXP writes the following message:

```
'S.LOAD 'CHECKSUM ERROR; [C,H,T]?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

If the record sequence number calculated by the loader does not equal the record sequence number in the last object record read, the DEXP writes the following message:

```
'S.LOAD ' SEQUENCE ERROR; [C,H,T]?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

If the filename specified in the Load command is not in the Diagnostic Facility Directory or is not entered in the Load command, the DEXP writes the following message:

```
* WARNING: FILE NOT FOUND IN DIRECTORY
```

Loading then continues with the DEXP issuing the following mount message prompt:

```
'handler' MOUNT 'progname' ON ttaa-dsa ; [C,H,T #,U]?
```

The previous message provides time to mount the appropriate magnetic tape or floppy diskette, and to ensure that the mechanism is ON and ready. If the progname is not on the IPL medium, the DEXP writes "SCRATCH" instead of the filename.

The DEXP writes the following load map message:

```
*LOADED:   'taskname'
 PGM:   address, length
```

If input and/or output buffers are assigned, the DEXP appends this message to the load map:

```
BFI:   address, length
BFO:   address, length
```

For more information, refer to the chapter on loading macrodiagnostic programs.

Because the taskname is substituted for the default program name (progname) specified in the object module, the DEXP writes the following message:

```
*WARNING:   TASK 'progname' NOW 'taskname'
```

3.4 Interrupt and Trap Processing

Traps and interrupts report asynchronous or synchronous events or requests to the Diagnostic Executive Program (DEXP). Generally, traps are routines that are synchronously entered if a fatal hardware error occurs. Interrupts are generally asynchronous if an I/O device requests service from the CPU.

The DEXP indentifies the source of each trap or interrupt automatically and responds to it according to its priority level. The highest priority interrupts for basic system integrity interrupts and traps, such as Power Fail-Safe/Auto-Start and Memory Parity Error. The next lower priority levels are for I/O service interrupts. External software interrupts use the lower level priorities. There are a total of 112 possible priority levels available. Each trap or interrupt has a dedicated Interrupt Vector Location which contains the 24-bit real address of its associated Interrupt Context Block.

The DEXP processes traps and interrupts in the Block Without Activate mode. In this mode, when the highest requesting level becomes the highest priority level, the CPU acknowledges the request and blocks all interrupts until receiving an Unblock command.

If a fatal error trap occurs, the fatal trap abort processor writes the following message to the operator communications (OC) file:

```
'tasknam1' ABORTED: abort code AT PSD: psdw1-psdw2 (cpusw)
'tasknam2' caused the abort
(abort code explanation)
```

tasknam1	specifies the one to eight character name of the source task being aborted.
tasknam2	specifies the one to eight character name of the task causing the abort (written if different from tasknam1).
abort code	specifies the code for the type of abort.
abort code explanation	specifies the description of the abort code.
psdw1-psdw2	specifies the contents of the PSD (program status doubleword) Word 1 and Word 2 at the point where the trap occurred.
cpusw	specifies the contents of the central processing unit status word which reflects the current operating condition of the processor.

If the Dump command is ON and a fatal error trap occurs, the DEXP writes a postmortem dump to the listed output (LO) file and an error message to the operator communications (OC) file.

3.5 System Integrity Traps and Major System Interrupts

The following are System Integrity Traps and Interrupts, their priority levels and trap abort codes:

Trap/Interrupt	Description	Relative Priority	Code
CPU/IPU:PWRD	Power Fail-Safe Trap	00	
CPU/IPU:PWRU	Auto-Start Interrupt	01	
CPU/IPU:MPAR	Memory Parity Error Trap	02	MP01
CPU/IPU:NMEM	Nonpresent Memory Error Trap	03	NM01
CPU/IPU:UNDF	Undefined Instruction Error Trap	04	UI01
CPU/IPU:PRIV	Privileged State Violation Trap	05	PV01
CPU/IPU:SVCT	Supervisor Call Trap	06	
CPU/IPU:MCHK	Machine Check Trap	07	MC01
CPU/IPU:SCHK	System Check Trap	08	SC01
CPU/IPU:MFLT	Map Fault Trap	09	MF01
IPU:IO*	IPU I/O Instruction Trap	0A	
CPU/IPU:SIPU*	SIPU Instruction Trap	0B	
CPU/IPU:ADDR	Address Specification Trap	0C	AS01
CPU/IPU:ATTN	Attention Interrupt	0D	
CPU/IPU:PRIH	Privilege "Halt" Trap	0E	HT01
CPU/IPU:ARIT	Arithmetic Exception Trap	0F	AE01
CPU/IPU:CACH	Cache Error Trap	10	CA01
RTCLOCK	Real-Time Clock Interrupt	18	
CPU/IPU:SPUR	Spurious Interrupt Receiver	NONE	

* These traps are valid only with CPU/IPU configurations.

If a system trap abort code is generated, the DEXP writes a program status word indicating where the trap occurred. The CPU status word defines the reason for the trap.

The following sections provide a description of these traps and interrupts.

3.5.1 Power Fail-Safe Trap

CPU/IPU:PWRD

If a system power failure occurs, the DEXP establishes auto-start linkage and halts the CPU. It makes no attempt to save I/O in progress.

3.5.2 Power Auto-Start Trap

CPU/IPU:PWRU

After system power is restored, the DEXP continues as a "warm start" by reinitializing all tasks, purging all device and file reservations, and issuing a DXP> prompt.

3.5.3 Memory Parity Error Trap

CPU/IPU:MPAR

A memory parity error is a fatal condition and the DEXP aborts the requesting task (MP01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.4 Nonpresent Memory Error Trap

CPU/IPU:NMEM

A nonpresent memory error is a fatal condition and the DEXP aborts the requesting task (NM01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.5 Undefined Instruction Error Trap

CPU/IPU:UNDF

Execution of an undefined instruction is a fatal condition and the DEXP aborts the requesting task (UI01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.6 Privileged State Violation Trap

CPU/IPU:PRIV

A privileged state violation is a fatal error and the DEXP aborts the requesting task (PV01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

DEXP is written in the Privileged mode, which allows the CPU to perform all its control functions and modify any part of the system. However, for any program written in the Unprivileged mode, memory protection is in effect and the system prohibits all privileged operations to that program.

3.5.7 Supervisor Call Trap

CPU/IPU:SVCT

A Supervisor Call instruction vectors through interrupt vector location which, in turn, points to 16 interrupt context blocks. In the DEXP, the active supervisor calls are as follows:

(SVC 0,0W)	Register and PSD stack push function (PUSH)
(SVC 1,0W)	Register and PSD stack pop function (POP)
(SVC 2,0W)	Round-robin task dispatcher (RETURN)
(SVC 3,0 through 31W)	Executive Service Call linkage
(SVC 4,0 through 31W)	Subroutine Call linkage
(SVC 5,0 through 31W)	Diagnostic Service Call linkage
(SVC 6 through SVC15)	Not used

3.5.8 Machine Check Trap

CPU/IPU:MCHK

Detection of a hardware or firmware failure is a fatal condition and the DEXP aborts the requesting task (MC01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.9 System Check Trap

CPU/IPU:SCHK

A software failure that tries to force the CPU into an illogical sequence is a fatal condition and the DEXP aborts the requesting task (SC01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.10 Map Fault Trap

CPU/IPU:MFLT

Detection of a map fault is a fatal condition and the DEXP aborts the requesting task (MF01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.11 I/O Instruction Trap

IPU:IO

The IPU executing an I/O instruction causes this trap.

3.5.12 SIPU Instruction Trap

CPU/IPU:SIPU

If the IPU executes a SIPU instruction, it interrupts the CPU. If the CPU executes a SIPU, it interrupts the IPU.

3.5.13 Address Specification Trap

CPU/IPU:ADDR

An address specification error is a fatal condition and the DEXP aborts the requesting task (AS01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes the abort to the task and the task returns to the point of interrupt, or continues processing, as appropriate.

3.5.14 Attention Interrupt — CPU/IPU:ATTN

Entering @@A on the console while in Console mode activates operator communications S.OPCOM. If operator communications is active and attention is requested, the DEXP ignores the request.

3.5.15 Privileged Halt Trap — CPU/IPU:PRIV

Execution of a privileged halt instruction is a fatal condition and the DEXP aborts the requesting task (HT01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes the abort to the task and the task returns to the point of interrupt, or continues processing, as appropriate.

3.5.16 Arithmetic Exception Trap — CPU/IPU:ARIT

Detection of an arithmetic exception is a fatal condition and the DEXP aborts the active task (AE01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.17 Cache Error Trap — CPU/IPU:CACH

Detection of a cache error is a fatal condition and the DEXP aborts the active task (CA01 abort code) unless the task expects and services the trap. If the task services the trap, the trap processor passes control to the task and the task returns to the point of the trap, or continues processing, as appropriate.

3.5.18 Real-Time Clock Interrupt — RTCLOCK

The Real-Time Clock interrupt updates the system elapsed time, provides a constant interval for timing I/O operations, and drives the time slice mechanism.

If a time slice interrupt occurs, Real-Time Clock interrupt processing first checks to see if a time slice is permitted by the currently running program. If a time slice is permitted, the DEXP pushes the task's registers and PSD and passes the requesting task to the round-robin task dispatcher (Supervisor Call 2) to be queued. If no time slice is permitted, the DEXP restores task's registers and returns to the point of interrupt.

3.5.19 Spurious Interrupt Receiver — CPU/IPU:SPUR

The spurious interrupt receiver responds to unexpected interrupts for which no interrupt handler exists. The spurious interrupt receiver counts an interrupt when it occurs and returns to the point of interrupt.

3.6 Executive Service (SVC) Abort Codes

The following is a list of the Gould CONCEPT/32 Diagnostic Executive Program executive service (SVC) abort codes:

Code	Description
E503	A program specifies an illegal executive service (secondary vector) for a supervisor call 3.
E504	A program specifies an illegal executive service (secondary vector) for a supervisor call 4.
E505	A program specifies an illegal executive service (secondary vector) for a supervisor call 5.
E506	A program specifies an illegal executive service (secondary vector) for a supervisor call 6.
E507-F	A program specifies an undefined executive service (supervisor calls 7 through 15).
LF01	A program specifies an undefined file code (SVC GET,PUT,CTRL).
LF02	A program specifies an undefined device type code (SVC GET,PUT,CTRL).
PS01	A PSD stack overflow occurs while attempting to PUSH (save) the task's PSD in the PSD stack (SVC PUSH).
PS02	A PSD stack underflow occurs while attempting to POP (remove) the task's PSD in the PSD stack (SVC POP).
RS01	A register stack overflow occurs while attempting to PUSH (save) the task's registers in the register stack (SVC PUSH).
RS02	A register stack underflow occurs while attempting to POP (remove) the task's registers from the register stack (SVC POP).
SP01	A length specification error occurs while reading (SVC READSPAD) or writing (SVC WRITSPAD) the system scratchpad (length specified is less than 1 or more than 256 words).
SP02	An address specification error occurs while reading (SVC READSPAD) or writing (SVC WRITSPAD) the system scratchpad (an address of less than 00_{16} or greater than FF_{16}).
SP03	The program detects an absent channel or device subaddress entry in the system scratchpad while attempting to perform a physical device assignment (no configured channel or device subaddress).
SP04	The program assigns an illegal I/O class designator in the system scratchpad while attempting to perform a physical device assignment (I/O class designator not 0, 1, 2, 3, E, or F).

Code	Description
SP05	The program detects an illegal scratchpad format while attempting to perform a physical device assignment (no continuity between device entry and interrupt entry).
SP06	The program detects an illegal scratchpad format while attempting to perform a physical device subaddress assignment (no continuity between device subaddress entry and interrupt entry).
TA01	The program attempts to activate a non-resident task.
TA02	The program attempts to activate a resident task at an entry point not defined by the task.
TA03	The program attempts to transfer control to a task outside the task's execution area.

3.7 Run-time Abort Codes

The following is a list of the Gould CONCEPT/32 Diagnostic Executive Program run-time abort codes:

Code	Description
RT01	A directive references an undefined run-time processor.
RT02	A jump directive references an undefined statement number.
RT03	A directive specifies an undefined letter modifier.
RT04	A directive specifies an undefined count modifier.
RT05	A directive specifies an undefined device model number.
RT06	An Assign directive specifies that more than 16 device subaddresses are assigned to a macrodiagnostic program.
RTks	A literal directive specifies an undefined or illegal keyword (k) or subfield (s).

3.8 I/O Handler Abort Codes

The following is a list of the Gould CONCEPT/32 Diagnostic Executive Program I/O handler abort codes:

Code	Description
DC01	The program attempts to assign a disc unit whose scratchpad device entry defines the device as other than I/O class E_{16} or I/O class F_{16}.
DC02	The program attempts to specify an illegal or undefined order to the disc driver.
DC03	The program attempts to process an Initialize Channel command to an IOP or RPU.
DC04	The disc handler declares an unrecoverable I/O error after exhausting the retry count.
DC05	The disc handler times out waiting for a service interrupt.
DC06	The program attempts to perform an Initialize Controller command.
DC07	The program attempts to perform a file operation with no blocking buffer.
LP01	The program attempts to assign a line printer whose scratchpad device entry defines the device as other than I/O class F_{16}.
LP02	The program attempts to specify an illegal or undefined order to the line printer driver.
LP03	The program attempts to process an Initialize Channel command to an IOP or RPU.
LP05	The line printer handler times out waiting for a service interrupt.
MT01	The program attempts to assign a magnetic tape unit whose scratchpad device entry defines the device as other than I/O class E_{16} or I/O class F_{16}.
MT02	The program attempts to specify an illegal or undefined order to the magnetic tape driver.
MT03	The program attempts to process an Initialize Channel command to an IOP or RPU.
MT04	The magnetic tape handler declares an unrecoverable I/O error after exhausting the retry count.

3.9 Compile-Time Abort Codes

The following is a list of the Gould CONCEPT/32 Diagnostic Executive Program directive compile-time abort codes:

Code	Description
CT01	The character immediately following a comma in a directive is not an alphabetic character or a number (undefined directive).
CT02	A statement number is not within the range 0 through 9 (statement number too large).
CT03	A directive count modifier is not within the range 0 through 32,767 (count modifier too large).
CT04	The character immediately following a directive is not a comma (missing comma).
CT05	A special character that follows a letter modifier is not an equal sign (syntax error).
CT06	The program detects a missing single quote while processing a literal directive.
CT07	The program detects and invalid operand while processing a parameter or jump directive.
CT08	The program detects an incorrectly specified parameter identifier or number while processing a jump or parameter directive.
CT09	The target of a jump is other than a statement number or a parameter Z.
CT10	The character immediately following the statement number or parameter Z in a jump directive is other than a slash or a comma.
CT11	The program detects a call sequence table overflow (too many directives in a directive string).
CT12	A program attempts to compile an undefined macro.
CT13	A macro attempts to execute itself. (Macro recursion is not supported.)

3.10 Programmed Halts

The Gould CONCEPT/32 Diagnostic Executive Program halts the processor under the following fatal error conditions:

- The CPU receives an unexpected trap during DEXP initialization.
- The device entry's class for the operator communications file within the scratchpad is other than 2 (TLC console) or F (IOP console).
- The CPU receives a power fail trap.
- The IPU receives a power fail trap.
- The IPU round-robin task dispatcher's forward link is destroyed.
- The CPU round-robin task dispatcher's forward link is destroyed.
- A task abort stack overflow occurs.
- The system receives a trap while S.DEXP is running.
- The teletype assignment is to a device other than class 2 (TLC console) or F (IOP console).
- The program receives the IOP console service interrupt without channel end status.
- The device class of the IOP console is other than class F.
- The IOP channel initialization fails.
- The TLC or IOP service interrupt occurs with a device class other than 2 (TLC) or F (IOP).
- The program returns an error status on the IOP console device.

The DEXP writes the program counter address causing the halt. To determine the reason for the halt, check the program listing at the indicated program counter address.

CHAPTER 4

DEXP COMMANDS

4.1 Command Overview

The DEXP processes commands that perform specific control and inquiry functions. These commands allow complete control over the system.

The commands are as follows:

ABORT	Terminates a task's execution.
ASSIGN	Assigns a physical device to a system file code or a unit under test to a program.
BATCH	Allows command file execution.
BIAS	Allows processor selection.
CONTINUE	Resumes a task previously suspended or held.
DEFINE	Defines a temporary macro.
DELETE	Deletes a temporary macro.
DUMP	Dumps specified memory, the scratchpad, the Task Communications Area (TCA), the Program Status Doubleword (PSD), and/or the registers to the listed output (LO) file.
EJECT	Advances a system file to a new page.
EXAMINE	Writes the contents of a specified memory location to the operator communications (OC) file.
EXIT or X	Returns control from the DEXP to the Diagnostic Overlay Loader.
GOTO	Diverts command file execution to the specified line number.
HELP	Provides information about commands, argument specifications, and abort codes.
HOLD	Suspends a task's execution.
LOAD	Loads a program into available memory.
MODIFY	Modifies the contents of a specified memory location.
MODIFY SPAD	Modifies the contents of a specified CPU scratchpad location.

NOTE	Writes messages to the LO file (if the LO file assignment is to the line printer) and echoes the messages to the OC file during the batch process.
PAGESIZE	Specifies the number of lines per page for the OC file.
REINITALIZE	Reinitializes a previously loaded program.
RESET	Resets specific program options or bits in specified memory locations.
RESTART	Restarts program execution without resetting the pass counter.
REWIND	Repositions a system file to beginning of media.
SEARCH	Searches a specified memory location for specific values.
SET	Sets program options or bits in specified memory locations.
SHOW	Displays information about macros, tests, options, command files, and the Diagnostic Facility Directory to the OC file.
SNAP	Writes the contents of specified memory locations to the OC file.
SNAP SPAD	Writes the contents of specified CPU scratchpad locations to the OC file.
START	Starts program execution.
STATUS	Writes the status information of specified tasks to the LO file.
TEST	Tests program options or bits in specified memory locations.
TIME	Writes the system elapsed time to the LO file.
UPSPACE	Advances a system file to a new line.
WAIT	Suspends command file execution or removes Interactive mode from concurrent task execution.

Note: Use all commands in Interactive mode or in Batch mode if the command is specified within a command file (except the Goto command which is invalid in Interactive mode).

4.2 Command Arguments

The following is a list of valid arguments to use with DEXP commands. (Refer to the individual command for a list of arguments which apply.)

Argument	Description
ADDRESS	Specifies a one to six digit hexadecimal physical memory address.
ANY	Indicates a task runs on both the CPU and IPU as available.
BFILE	Indicates a level eight (batch) filename.
BIT #	Indicates a keyword specifying a bit operation. The # specifies a one or two-digit bit number.
CDATA	Specifies information for user terminals.
COUNT	Specifies a decimal number corresponding to the number of like operations to be performed or, for the Page command, specifies the number of lines per page.
CPU	Indicates a task runs on the central processing unit only.
DCB	Indicates the keyword device control block.
DDATA	Specifies information for hard platter discs.
DELAY	Indicates the length of a pause (in seconds) between nonmaindiagnostic tests used in the Start and Restart commands.
DSA	Specifies a two-character hexadecimal device subaddress.
FDT	Indicates the keyword file destination table.
FILE	Indicates a keyword specifying a file operation.
FORWARD	Indicates tests run in forward order.
INTERRUPT	Indicates a display of the interrupt count.
IOEA	Indicates the keyword I/O emulation area.
IPU	Indicates a task runs on the internal processing unit only.
LEVEL	Specifies a file type.
LFC	Specifies one of the following logical file codes: BI, BO, CC, LO, OC, SI, SO.
MACNAME	Specifies a macro name as defined in the task's macro directory.
MACRO	Indicates a keyword which specifies a macro-directive language operation.

Argument	Description
MASK	Specifies a one to eight digit hexadecimal quantity to be used as a bit mask when performing a memory search.
OFILE	Indicates a level two filename.
OFF	Indicates a keyword disabling a task memory dump upon abort or a system status display every 10 minutes.
ON	Indicates a keyword enabling a task memory dump upon abort or a system status display every 10 minutes.
OPTION #	Indicates a keyword specifying an option operation. The # specifies a one or two-digit option number.
PASS	Specifies the maximum number of passes for a macrodiagnostic program to run.
PSDS	Indicates the keyword task PSD stack.
RANDOM	Indicates tests run in random order.
RANGE	Specifies the limits of a command operation either as a single number or the limits between two numbers. Specify tests, options, bits, and lines in decimal, and addresses as one to six digit hexadecimal physical memory locations.
REGS	Indicates the keyword task register stack.
RELOCATE	Indicates the command relocates the specified addresses.
REVERSE	Indicates tests run in reverse order.
SPAD	Indicates the keyword CPU scratchpad memory.
TASK	Specifies a system task or macrodiagnostic program.
TCA	Indicates the keyword task control area.
TTAA	Specifies a device mnemonic, where TT specifies a two-character device label and AA specifies a two-character channel address.
TYPE-AHEAD	Indicates the following information answers task prompts in advance.
VALUE	Specifies a one to eight digit hexadecimal quantity which is right justified in relation to a two or three bit memory word.

4.3 Command Syntax

A command consists of a command verb and a number of arguments defined by the command verb syntax.

Syntax

DXP>verb arg_1 [,...arg_n]

verb — specifies a command.

arg — specifies required or optional arguments defined by the command verb syntax.

Some arguments contain either a keyword or a keyword with an associated subfield. A subfield modifies a keyword and if a subfield is present, the keyword/subfield combination is an argument. Delineate subfields with parentheses.

Syntax

dirtv='$kw_1(sf_1,...,sf_n),...,kw_n$'

kw — specifies a keyword.

sf — specifies a subfield.

The following rules apply when using commands:

- The DEXP accepts four-character command abbreviations.
- The DEXP accepts either upper and lower case alphabetic characters.
- The DEXP accepts only the first eight characters of an argument.
- The DEXP treats spaces and/or commas as argument delimiters.

4.4 Editing Commands

If an error is detected prior to pressing the Return key, eliminate an erroneous character by entering a backspace character, a Control-H, or a backspace arrow and reentering the desired character. Perform multiple deletions with successive backspacing.

Erase the entire command by entering a delete (DEL) or rub-out (RUB) character. The DEXP responds to a delete or rub-out with a carriage return, a line feed, and a new prompt.

4.5 Type-Ahead Function

The type-ahead function supplies, in advance, data required for execution. This data is particular to individual program execution. (Refer to each program description for possible type-ahead responses.) The DEXP stores the data in type-ahead buffers until required for processing. Without the type-ahead function, use a prompt/response sequence to provide this data.

Use type-ahead capabilities in the following three cases:

Data required before execution

For example, supply answers to prompts associated with a program's initialization sequence by using the type-ahead argument specified in either the Load or Reinitialize command.

Data required during execution

For example, supply answers to prompts associated with operator intervention tests or a constant error if Option 2 (halt on error) is set by using the type-ahead argument specified in either the Start or Restart command.

Data required during execution of a directive

For example, supply answers to prompts associated with the execution of the LMC directive by using the type-ahead argument specified in the directive syntax.

Specify type-ahead data in blocks of no more than 70 characters. Precede these blocks with an "at" sign (@) and terminate them with a semicolon. Delimit the type-ahead data entries with semicolons.

If used in command syntax, specify the command, the type-ahead block, and all other arguments on the same line as the command. Therefore, only one type-ahead block may be specified.

If used in directive syntax, a maximum of two type-ahead blocks may be specified. Specify the first block on the line containing the directive and the second block on the next line. If specifying two type-ahead blocks, enter a comma between the directive and its associated type-ahead data. This signifies continuation on a second line of the same directive string. Also, precede each type-ahead block with an "at" sign (@).

4.6 ABORT Command

This command terminates a task's execution.

Syntax

```
DXP>ABORT taskname
```

taskname specifies the one to eight character name of the program being aborted.

If the task being aborted is a macrodiagnostic program, the DEXP writes the following message to the listed output (LO) file:

```
hh:mm:ss 'taskname' ABORTED
```

Example

```
DXP>ABOR IP.FDD.r
```

This command aborts the Gould IOP Floppy Disc Diagnostic.

4.7 ASSIGN Command

This command assigns a physical device to a system file or to a macrodiagnostic program for testing.

4.7.1 File Assignment Derivatives

Syntax

DXP>ASSIGN lfc=ttaa [,dsa] [,ccc,MODEL(nnnn)] [,BAUD(#)] [,FDX / ,HDX]

lfc — specifies the two-character system file code for the DEXP system file being assigned. (BI, BO, CC, LO, OC, OL, SI, SO)

ttaa — specifies the four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.

dsa — specifies the two-character hexadecimal device subaddress.

ccc — specifies the channel type for disc initialization. Valid channel types are RPU and IOP. Specify this parameter only for hard platter discs.

MODEL(nnnn) — indicates the disc model number. Specify this parameter only for hard platter discs.

BAUD(#) — indicates the data transfer rate in bits per second if assigning the operator communications (OC) file to a user terminal. The specified baud rate must correspond to the baud rate of the assigned terminal. If the argument is not specified, the program assigns the default values: 9600 baud and Full-Duplex mode.

FDX — indicates Full-Duplex mode.

HDX — indicates Half-Duplex mode.

Example

```
DXP>ASSI LO=LP18
```

This command assigns a line printer channel address 18_{16} to the listed output (LO) file. If no device subaddress is specified, the program assumes a device subaddress of 00_{16}.

4.7.2 Program Assignment Derivatives

Syntax

DXP>ASSIGN taskname,ttaa, dsa_1[,...,dsa_{16}]

taskname specifies the one to eight character name of the program being assigned a physical device.

dsa specifies two-character hexadecimal device subaddresses.

ttaa specifies the four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.

The following are valid device type codes:

TY	Teletypewriter
LP	Line Printer
CR	Card Reader
MT,M9,M7	Magnetic Tape
FL	Floppy Disc
DC,DM	Hard Platter Disc
UT	User Terminal
CA	Remote Communications Adapter

Notes: Assign the operator communications (OC) file only to a teletypewriter (TY), a user terminal (UT), or remote communications adapter (CA).

Example

```
DXP>ASSI IP.FDD.r,FL7E,F0,F1
```

This command assigns a floppy disc channel address of $7E_{16}$ to the Gould IOP Floppy Disc Diagnostic. In addition, the command designates the device subaddresses $F0_{16}$ and $F1_{16}$ as the units under test.

4.8 BATCH Command

This command allows a command file to be accessed and executed. The Batch command may be used in the command file only as the last command. This links command files together allowing multiple files to be processed as a single file.

Syntax

DXP>BATCH filename [,range$_1$,...,range$_n$]

filename specifies the one to eight character name of the command file being executed.

range specifies an individual line number or a range of line numbers (one through 256) separated by a hyphen if executing an inclusive block of lines. To execute the entire command file, specify no range.

If the specified file is not a command file, the DEXP writes the following message to the OC file:

```
*FILE SPECIFIED NOT A BATCH FILE (LEVEL 8)
```

If the range specified is not between one and 256, the DEXP writes the following message to the OC file:

```
*VALID LINE NUMBERS ARE 1-256
```

Note: If the command control (CC) file is assigned to a null (NO) device, this command may not be used.

Example

```
DXP>BATC YOURFILE, 10-100
```

This command automatically executes lines 10 through 100 of the command file "YOURFILE".

4.9 BIAS Command

This command selects which processor runs a program.

Syntax

DXP>BIAS taskname [,CPU | ,IPU | ,ANY]

taskname specifies the one to eight character name of the program being biased.

CPU indicates that the program runs on the CPU only.

IPU indicates that the program runs on the IPU only.

ANY indicates that the program runs on either the CPU or the IPU as available.

Note: If the CPU or IPU argument is specified, DEXP executes this task on the selected processor until specifying a different processor.

Example

```
DXP>BIAS IP.FDD.r, IPU
```

This command runs the Gould IOP Floppy Disc Diagnostic on the IPU only, but I/O is still done by the CPU.

4.10 CONTINUE Command

This command resumes execution of a task suspended by a Hold command.

Syntax

```
DXP>CONTINUE taskname
```

taskname specifies the one to eight character name of the program being continued.

If the task being continued is a program, the DEXP writes the following message to the listed output (LO) file:

```
hh:mm:ss 'taskname' CONTINUED
```

Example

```
DXP>CONT S.LOAD
```

This command continues a suspended load process.

4.11 DEFINE Command

This command creates a macro routine containing directives. A macro defined with this command is a temporary macro (created and defined after program assembly). The program also contains permanent macros (assembled into the program) which cannot be redefined or deleted.

Syntax

```
DXP>DEFINE taskname,MACRO,mname
```

taskname specifies the one to eight character name of the program being assigned a macro.

MACRO indicates that the following is a macro.

mname specifies the one to eight character name of the macro being defined.

The DEXP then issues the macro prompt MAC>, awaits completion of the macro directive string, and reissues its DXP> prompt.

After creating a macro, the DEXP writes the following message to the operator communications (OC) file:

```
DEFINE MACRO 'mname' (xxx BYTES OF TEMPORARY STORAGE ARE AVAILABLE)
```

mname specifies the one to eight character name for referencing the macro.

xxx specifies the number of bytes available for temporary macro storage.

If an error occurs, the DEXP writes one of the following messages describing the error:

```
*DIRECTIVE STRING IS TOO LONG
*TASK DOES NOT SUPPORT MACRO(S)
*MACRO IS ALREADY DEFINED
*NO BUFFER FOR TEMPORARY MACRO(S)
*IMPROPER STRING DETECTED
*MACRO TABLE IS FULL
*MACRO NAME NOT ALPHANUMERIC
*MACRO NAME MUST BEGIN WITH "A-Z"
```

Example

```
DXP>DEFI JAK,MAC,DUMPX
MAC>PW=128,DI,DO
```

This command defines a temporary directive macro with the macroname DUMPX for the program JAK. The macro contains a directive string that sets parameter W equal to 128 then dumps the contents of the program's input and output buffers to the listed output (LO) file.

To execute a macro, enter the taskname and the macroname as follows:

```
DXP>JAK DUMPX
```

4.12 DELETE Command

This command deletes a temporary macro previously created using the Define command. Deleting the macro releases previously allocated buffer space for subsequent macro definitions. (Permanent macros cannot be deleted or redefined.)

Syntax

```
DXP>DELETE taskname, MACRO, mname
```

taskname specifies the one to eight characters name of the program assigned a macro file.

MACRO indicates that the following is a macro.

mname specifies the one to eight character name of the macro being deleted.

After deleting the macro, the DEXP writes the following message to the operator communications (OC) file:

```
'taskname' WHAT NEXT?
```

If an error occurs, the DEXP writes one of the following messages describing the error:

- *CAN'T DELETE PERMANENT MACRO(S)
- *MACRO IS UNDEFINED
- *MACRO NOT FOUND
- *TASK DOES NOT SUPPORT MACRO(S)

Example

```
DXP>DELE JAK,MAC,DUMPX
```

This command deletes the temporary macro DUMPX for the task JAK allowing the buffer space to be reallocated.

4.13 DUMP Command

This command writes the contents of memory (up to 16 megabytes), the scratchpad, the task communications area (TCA), the program status doubleword (PSD) stack, or the register (REG) stack to the listed output (LO) file. Dumps are either zero relative (from the system's first memory location) or task relative (from the task's first memory location). This command also enables or disables a postmortem dump for fatal trap processing.

4.13.1 Executive Dump Task (S.DUMP)

Issuing a Dump command activates the DEXP Executive Dump Task (S.DUMP) and writes the contents of specified locations to the listed output (LO) file in side-by-side ASCII-coded hexadecimal and ASCII format. Dumps may suppress like data.

If a task requests a dump, the DEXP dynamically suspends the calling task and the Dump Task runs in its place. If operator communications requests a dump, the Dump Task runs concurrently with operator communications.

The Dump Task runs in the Extended Addressing mode making it possible to dump any location within 16 megabytes of memory. All address arguments are 24-bit addresses. The smallest unit to be dumped is one word of data.

4.13.2 Dump Output

During the dump, the Dump Task verifies that the listed output (LO) file assignment is to a null device, a teletypewriter, or a line printer.

If the LO file is assigned to a null device, the Dump Task executes as if it were assigned to a peripheral device, except it displays nothing. This feature makes it possible to verify the contents of physical memory for parity errors without actually performing the display.

If the LO file is assigned to a teletypewriter, the Dump Task sets a four-word boundary and displays the data four items per line. The Dump Task does not reserve the device, allowing access to the system after a dump begins. Therefore, a dump can be aborted after finding the required information.

If the LO file is assigned to a line printer, the Dump Task set an eight-word boundary and displays data eight items per line. The Dump Task reserves the line printer to allow a continuous display of data.

4.13.3 Zero Relative Dump Derivative

Syntax

```
DXP>DUMP [, range]
```

range — specifies a single memory address or a range of memory addresses separated by a hyphen if dumping an inclusive block of memory addresses.

Note: If no address specification is made, the command dumps all memory locations within physical memory. If only one address is specified, the command dumps that memory location.

Example

```
DXP>DUMP 700-7FC
```

This command dumps the contents of memory locations 700_{16} through $7FC_{16}$ to the listed output (LO) file.

4.13.4 Task Relative Dump Derivative

Syntax

```
DXP>DUMP taskname [,range]
```

taskname — specifies the one to eight character name of a program whose addresses are being dumped.

range — specifies a single memory address or a range of memory address separated by a hyphen if dumping an inclusive block of memory addresses.

Note: If no address specification is made, the command dumps all memory locations associated with the task. If only one address is specified, the command dumps that memory address.

Example

```
DXP>DUMP IP.FDD.r,50-FC
```

This command dumps the contents of memory locations 50_{16} through FC_{16} relative to the Gould IOP Floppy Disc Diagnostic.

4.13.5 PSDS, REGS, and TCA Dump Derivatives

Syntax

DXP>DUMP taskname [,PSDS / ,REGS / ,TCA]

taskname specifies the one to eight character name of the program whose PSDS, REGS, or TCA is being dumped.

PSDS indicates dumping the program's status doubleword (PSD) stack.

REGS indicates dumping the program's register (REG) stack.

TCA indicates dumping the program's task communications area (TCA).

Example

```
DXP>DUMP IP.FDD.r,REGS
```

This command dumps the contents of the register stack within the Gould IOP Floppy Disc Diagnostic.

4.13.6 Scratchpad Dump Derivative

Syntax

DXP>DUMP SPAD [,range]

SPAD indicates dumping a portion of the system scratchpad.

range specifies a single memory address or a range of memory addresses separated by a hyphen if dumping an inclusive block of memory addresses.

Note: If no range is specified, the command dumps all 256 words (00_{16} through FF_{16}) of the scratchpad. If only one address of the range is specified, the command dumps that scratchpad location.

Example

```
DXP>DUMP SPAD,20-2F
```

This command dumps the contents of the system scratchpad locations 20_{16} through $2F_{16}$.

4.13.7 Postmortem Dump Controls

Syntax

```
DXP>  DUMP  { ON  }
            { OFF }
```

ON indicates that the command enables the postmortem dump and writes a dump upon the occurrence of a fatal error abort trap.

OFF indicates that the command disables the postmortem dump and does not write a dump upon the occurrence of a fatal error abort trap. By default, the postmortem dump is disabled.

4.13.8 Directory Derivative

Syntax

```
DXP>DUMP DIRECTORY
```

DIRECTORY indicates a dump of the Diagnostic Facility Directory file names.

4.13.9 FDT, DCB, IOEA Dump Derivatives

Syntax

```
DXP>DUMP { ,FDT                  }
         { ,DCB  [,ca [,dsa]]    }
         { ,IOEA                 }
```

FDT indicates dumping the contents of the file definition tables (FDT).

DCB indicates dumping the contents of the device control blocks (DCB).

ca specifies the hexadecimal channel address of the DCB if dumping only one DCB.

dsa specifies the hexadecimal device subaddresses of the DCB if dumping only one DCB.

If a channel address is not specified, the command dumps all DCBs.

IOEA indicates dumping the contents of the I/O emulation area (IOEA).

4.14 EJECT Command

This command causes the device assigned to a system file to begin printing at the top of a page.

Syntax

```
DXP>EJECT lfc [,ct]
```

lfc — specifies the two-character system file code of the operator communications (OC) or listed output (LO) file being ejected.

ct — specifies the number of pages to eject. If the count is not specified, the command ejects one page.

Example

```
DXP>EJEC LO,2
```

This command ejects two pages on the physical device assigned to the listed output (LO) file.

4.15 EXAMINE Command

This command writes the contents of a specified range of memory locations to the operator communications (OC) file. Memory ranges are either task relative (from the task's first memory location) or zero relative (from the system's first memory location).

Syntax

DXP> { SPAD / EXAMINE } taskname ,range$_1$ [,...,range$_n$]

taskname — specifies a one to eight character name of the task if the address being examined is task relative.

SPAD — indicates that the CPU's scratchpad memory is being examined.

range — specifies a single memory address or a range of memory addresses separated by a hyphen if examining an inclusive block of memory addresses. The scratchpad address range is 00_{16} through FF_{16}. If the SPAD argument is specified, all ranges are optional. However, if the memory addresses to be examined are task relative, at least one range must be specified.

Example

```
DXP>EXAM IP.FDD.r,100
```

This command writes the contents of memory location 100_{16}, relative to the base address of the Gould IOP Floppy Disc Diagnostic, to the OC file.

4.16 EXIT Command

This command returns control from the DEXP to the Diagnostic Overlay Loader. If the DEXP is started as a result of command file execution processing initiated by the Diagnostic Overlay Loader BATCH command, command file processing resumes from the point the DEXP exits. If not, the Diagnostic Overlay Loader issues its prompt.

Syntax

```
DXP>EXIT
```

4.17 GOTO Command

This command causes command file processing to branch from its current position to the specified line number within the command file.

Note: This command cannot execute in Interactive mode.

Syntax

```
DXP>GOTO linenumber
```

linenumber specifies the next command file line number to be executed. This line number corresponds to the relative record number of a line within the command file.

If the specified line number is invalid, the DEXP writes the following message to the DC file:

```
•VALID LINE NUMBERS ARE 1-256
```

Example

```
DXP>GOTO 50
```

This command branches from the current position to line 50 to continue execution.

4.18 HELP Command

This command provides information about command syntax, argument usage, and abort conditions.

Syntax

DXP>HELP { abortcode$_1$ [,...,abortcode$_n$] / argument$_1$ [,...,argument$_n$] / command$_1$ [,...,command$_n$] }

abortcode — specifies the DEXP abortcode on which help is requested.

argument — specifies the DEXP argument on which help is requested

command — specifies the DEXP command on which help is requested.

Example

```
DXP>HELP START
```

This command writes the following Start command syntax to the OC file:

```
DXP>STAR TASKNAME [,RANGE,...][FOR,REV,RAN,CPU,IPU,ANY,PASS(#),DELAY(#,#,P)]
DXP>STAR TASKNAME... [@ [TYPA]]
```

*NOTE: ARGUMENTS ENCLOSED BY PARENTHESES MUST BE ENCLOSED BY PARENTHESES. ARGUMENTS ENCLOSED BY BRACKETS ARE OPTIONAL.

4.19 HOLD Command

This command suspends a task's execution until entering a Continue command.

Syntax

```
DXP>HOLD taskname
```

If the task being held is a program, the DEXP writes the following message to the listed output (LO) file:

```
hh:mm:ss 'taskname' HELD
```

Example

```
DXP>HOLD S.LOAD
```

This command suspends the load process until entering a Continue command.

4.20 LOAD Command

This command causes the DEXP to load a task from the binary input (BI) file into available memory.

Syntax

DXP>LOAD [taskname] [(filename)] [,address] [@type$_1$; ... ;type$_n$;]

taskname specifies the one to eight character name given to the task.

(filename) specifies the eight-character name of the level two macrodiagnostic program corresponding to the Diagnostic Facility Directory entry. The command syntax requires that the argument be enclosed in parentheses.

address specifies an load address if the task is to be loaded at other than the default program counter address.

@ indicates that the following is type-ahead data.

type specifies the type-ahead data. Delimit individual type-ahead entries with a semicolon.

Note: If the filename is not in the Diagnostic Facility Directory, the DEXP writes the following message:

• WARNING: FILE NOT FOUND IN DIRECTORY

The DEXP then issues the following prompt:

'handler 'MOUNT TASK' progname' ON ttaa-dsa; [C,H,T,#,U] ?

progname specifies the eight-character name corresponding to the directory entry. (If the file is not on the IPL tape, the DEXP writes "SCRATCH" instead of the filename.)

ttaa specifies the four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.

dsa specifies the two-character hexadecimal device subaddress.

If loading from the IPL device, respond with C to continue execution, with H to continue suspension, or with T to terminate execution. If loading from other than the IPL device, respond with the relative file position, the coordinates of the file (#), or with U (specifying an unblocked file). The DEXP loads the file and continues execution.

Example

```
DXP>LOAD JAK(IP.MPI.r), 10000 @ML;;30;40;
```

This command loads the Gould IOP Multipurpose Custom Interface Diagnostic (IP.MPI.r) under the taskname JAK beginning at location 10000_{16} and initializes it with type-ahead data.

4.21 MODIFY Command

This command modifies the contents of a memory location by replacing it with a specified hexadecimal value. The DEXP echoes the new contents to the operator communications (OC) file.

Syntax

DXP>MODIFY [taskname] ,range$_1$(value [,R]) [,...,range$_n$(value [,R])]

taskname — specifies the one to eight character name of the task if the range specified is task relative (from the task's first memory location). Otherwise, the argument is optional.

range — specifies a single memory address or a range of memory addresses separated by a hyphen if modifying an inclusive block of memory addresses.

value — specifies the hexadecimal value that replaces the contents of the specified memory addresses.

R — indicates that the value given is being biased by the task's base address prior to modification.

Example

```
DXP>MODI  IP.FDD.r,104(FFFFFFFF)
```

This command modifies the contents of memory location 104_{16} in the Gould IOP Floppy Disc Diagnostic to the value $FFFFFFFF_{16}$.

4.22 MODIFY SPAD Command

This command modifies the contents of a specified CPU scratchpad location by replacing it with a specified hexadecimal value. The DEXP echoes the new contents to the operator communications (OC) file. This command does not affect the IPU.

Note: The DEXP does not attempt to validate the scratchpad location being modified.

Syntax

DXP>MODIFY SPAD,range$_1$(value) [,...,range$_n$(value)]

SPAD — indicates the system scratchpad memory is being modified.

range — specifies a single memory address or a range of memory addresses separated by a hyphen if modifying an inclusive block of memory addresses. Valid ranges are 0 through FF.

value — specifies the hexadecimal value that replaces the contents of the specified range of memory addresses.

Example

```
DXP>MODI  SPAD,7E(0F6C7E00)
```

This command modifies the contents of scratchpad location $7E_{16}$ to the value $0F6C7E00_{16}$.

4.23 NOTE Command

This command adds comment lines to a command file. The DEXP writes any message following the Note command to the listed output (LO) file (if the LO assignment is to the line printer) and echoes it to the operator communications (OC) file during batch processing.

Syntax

```
DXP>NOTE message
```

message specifies the comment lines being added to the command file.

Example

```
DXP>NOTE EXAMPLE NOTE COMMAND
```

This command writes the message "EXAMPLE NOTE COMMAND" to the LO file, echoes it to the OC file, and adds it to the command file.

4.24 PAGESIZE Command

This command sets the length of one page of output for the operator communications (OC) file.

Syntax

```
DXP>PAGESIZE n
```

n specifies the number of lines required for one page of output. The maximum number of lines is 24. By default, the page size is set to zero.

Example

```
DXP>PAGE 18
```

This command sets the length of one page on the operator communications (OC) file to 18 lines. Each time the DEXP writes one page of data to the OC file, it also writes the following message:

```
ENTER CR FOR MORE
```

Responding with the Return key results in additional lines being written. Responding with any character other than the Return key results in output termination.

CAUTION

If using the Pagesize command, ensure that an uncancelled prompt does not cause tasks to back up waiting to write to the operator communications (OC) file.

4.25 REINITIALIZE Command

This command reinitializes a program to its initial load conditions.

Syntax

DXP> REINITIALIZE taskname [,CPU / ,IPU / ,ANY] [@typa$_1$; [...;typa$_n$;]]

taskname specifies the one to eight character name of the program being reinitialized.

CPU indicates that the task runs on the central processing unit only.

IPU indicates that the task runs on the internal processing unit only.

ANY indicates that the task runs on either the CPU or the IPU as available.

@ specifies that the following is type-ahead data.

typa specifies the type-ahead data. Delimit individual type-ahead entries with a semicolon.

Note: If the CPU or IPU argument is specified, the DEXP executes this task on the selected processor until the task is reinitialized again, specifying a different processor.

Use this command to perform a system cold start by entering CN.DXP.r as the taskname if no programs are active in the system. If programs are active and an attempt is made to reinitialize CN.DXP.r, the DEXP writes the following message to the operator communications (OC) file:

```
*DIAGNOSTIC ACTIVE: COMMAND IGNORED
```

Example

```
DXP>REIN IMP.MPI.r, CPU @ML;;30;40;
```

This command reinitializes the Gould IOP Multipurpose Custom Interface (IP.MPI.r) Diagnostic. The task runs on the CPU and reinitializes with the specified type-ahead data.

4.26 RESET Command

This command resets program options or bits in a specified memory location.

4.26.1 Option Word Derivative

Syntax

DXP>RESET [taskname],OPT ,opt_1 [,...,opt_n]

taskname specifies the one to eight character name of the program whose options are being reset. If no taskname is specified, this command resets the Diagnostic Executive Program (CN.DXP.r) options.

OPT indicates that options are being reset.

opt specifies the options being reset.

Example

DXP>RESE 1P.FDD.r,OPT,8,11

This command resets Options 8 and 11 in the Gould IOP Floppy Disc Diagnostic.

4.26.2 Bit Manipulation Derivative

Syntax

DXP>RESET [taskname] ,address,BIT,bit_1 [,...,bit_n]

taskname specifies the one to eight character name of the program whose bits are being reset. If no taskname is specified, this command resets the specified bit in memory.

address specifies the hexadecimal address of the word whose bits are being reset.

BIT indicates that bits are being reset.

bit specifies the bits being reset.

Example

DXP>RESE 78FC,BIT,1,4

This command resets bits 1 and 4 at memory location $78FC_{16}$.

4.27 RESTART Command

This command restarts program execution without resetting the pass counter.

Syntax

DXP>RESTART taskname [,CPU / ,IPU / ,ANY] [,REV / ,RAN / ,FOR] [,range$_1$[,...,range$_n$]] [,PASS(n)] [,DELAY(t,p [,P])] [@typa$_1$;[...;typa$_n$;]]

taskname — specifies the one to eight character name of the program being restarted.

CPU — indicates that the task runs on the central processing unit only.

IPU — indicates that the task runs on the internal processing unit only.

ANY — indicates that the task runs on either the CPU or IPU as available.

REV — indicates that the selected tests run in reverse order.

RAN — indicates that the selected tests run in random order.

FOR — indicates that the selected tests run in forward order.

range — specifies an individual test number or range of test numbers, separated by a hyphen if executing an inclusive block of tests. END may be specified instead of the last test number as the second parameter of the range. If no range is specified, the program executes all tests.

PASS(n) — indicates the number of passes the task runs.

DELAY — indicates performing a delay between tests and/or passes as specified by the subfields.

- t — specifies, in seconds, the inter-test delay interval beginning upon completion of each test.
- p — specifies, in seconds, the inter-pass delay interval beginning upon completion of each pass.

  Note: The subfields in the delay argument are order dependant and therefore, zero (null value) or some valid time value must be specified for each delay interval (inter-test and inter-pass) parameter.
- P — indicates the subfield entry specifying that the preceding delay intervals are intra-test/pass intervals.

@ — specifies that the following is type-ahead data.

typa — specifies the type-ahead data. Delimit individual type-ahead entries with a semicolon.

Note: If reverse (REV) or random (RAN) order is specified in the Restart command of a program which does not support these functions, the program runs in forward sequential order and writes the following message to the OC file:

```
*TASK DOESN'T ALLOW NON-SEQUENTIAL TESTING
```

Use this command to perform a system warm start by entering CN.DXP.r as the taskname if no programs are active in the system. If programs are active and an attempt is made to restart CN.DXP.r, the DEXP writes the following message to the operator communications (OC) file:

```
*DIAGNOSTIC ACTIVE: COMMAND IGNORED
```

Examples

```
DXP>REST IP.FDD.r
```

This command restarts the Gould IOP Floppy Disc Diagnostic and runs all tests.

```
DXP>REST IP.FDD.r,3-7,14-27
```

This command restarts the Gould IOP Floppy Disc Diagnostic and runs only Tests 3 through 7, and Tests 14 through 27.

```
DXP>REST IP.FDD.r, CPU,REV,PASS(10)
```

This command restarts the Gould IOP Floppy Disc Diagnostic. The program runs 10 passes on the CPU in reverse order.

```
DXP>REST IP.FDD.r,IPU,DELAY(10,P)@C;
```

This command restarts the Gould IOP Floppy Disc Diagnostic. The task runs on the IPU only with an intra-test delay interval of 10 seconds and automatically responds to the first prompt with C to continue program execution.

4.28 REWIND Command

This command repositions a file to the beginning of media.

Syntax

```
DXP>REWIND lfc
```

lfc specifies one of the following two-character system file codes being rewound:

OC	Operator Communications File
LO	Listed Output File
BI	Binary Input File
BO	Binary Output File
SI	Scratch Input File
SO	Scratch Output File

Example

```
DXP>REWI BI
```

This command rewinds the binary input (BI) file.

4.29 SEARCH Command

This command searches the contents of specified memory locations for a value (optionally masked) and writes the results of each successful comparison to the operator communications (OC) file.

Syntax

DXP>SEARCH[taskname],$range_1$(value[,mask]) [,...,$range_n$(value[,mask])]

taskname specifies an one to eight character name of the program being searched.

range specifies a single memory address or a range of memory addresses separated by a hyphen if searching an inclusive block of memory.

value specifies the hexadecimal value which is the object of the search.

mask specifies the optional mask pattern if other than $FFFFFFFF_{16}$.

If only one address is specified, the function is identical to an Examine command. If only the first and last addresses are specified, the function is identical to a Snap command.

Example

```
DXP>SEAR 78FC-8112(01010101)
```

This command searches memory locations $78FC_{16}$ through 8112_{16} for the value 01010101_{16} using the mask $FFFFFFFF_{16}$.

4.30 SET Command

This command sets program options or bits in a specific memory location.

4.30.1 Option Word Derivative

Syntax

DXP>SET [taskname],OPT,opt_1 [,...,opt_n]

taskname specifies the one to eight character name of the program whose options are being set. If no taskname is specified, this command sets the Diagnostic Executive Program (CN.DXP.r) options.

OPT indicates that options are being set.

opt specifies the options being set.

Example

DXP>SET IP.FDDr,OPT,2

This command sets Option 2 (halts on error) in the Gould IOP Floppy Disc Diagnostic.

4.30.2 Bit Manipulation Derivative

Syntax

DXP>SET [taskname] ,address,BIT,bit_1 [,...,bit_n]

taskname specifies the one to eight character name of the program whose bits are being set. If no taskname is specified, this command resets the specified bit in memory.

address specifies the hexadecimal relative address of the word whose bits are being set.

BIT indicates that bits are being set.

bit specifies the bits being set.

Example

DXP>SET IP.FDD.r,100,BIT,1,31

This command sets bits 1 and 31 in memory location 100_{16} in the Gould IOP Floppy Disc Diagnostic.

4.31 SHOW Command

This command displays information about macros, tests, options, command files, and the Diagnostic Facility Directory to the operator communications (OC) file.

4.31.1 Macro Derivative

This command writes a list of all currently defined macros, (if no macroname is specified) or a list of the directives in a specified macro to the OC file.

Syntax

```
DXP>SHOW taskname, MACRO [,mname]
```

taskname specifies the one to eight character name of the program whose macro is being shown.

MACRO indicates that macro files are being shown.

mname specifies the one to eight character macro name.

If MACRO is specified with a task which does not support macros, the DEXP writes the following message to the OC file:

```
*TASK DOES NOT SUPPORT MACRO(S)
```

If the specified macro is undefined, the DEXP writes the following message to the OC file:

```
*MACRO NOT FOUND
```

4.31.2 Options Derivative

This command writes the program option set to the operator communications (OC) file.

Syntax

```
DXP>SHOW [taskname] OPT
```

taskname specifies the one to eight character name of the program whose options are being shown. If no taskname is specified, this command shows the Diagnostic Executive Program (CN.DXP.r) options.

OPT indicates that options are being shown.

4.31.3 Show Directory Derivative

This command writes a list of the program names in the Diagnostic Facility Directory.

Syntax

```
DXP>SHOW DIRECTORY
```

4.31.4 Test Derivative

This command writes a list of the specified test's directives to the operator communications (OC) file (valid only for directive tests).

Syntax

DXP>SHOW taskname, TEST [,test$_1$ [,...,test$_n$]]

taskname specifies the one to eight character name of the program whose tests are being shown.

TEST indicates that tests are being shown.

test specifies the test numbers whose contents are being shown. The tests must be directive tests. If no test numbers are specified, the command writes a list of test titles.

If tests are specified with a task which does not have tests, the DEXP writes the following message to the OC file:

•TASK DOESN'T HAVE TESTS

If the specified test is not a directive test, the DEXP writes the following message to the OC file:

•IN-LINE CODED TEST

If the specified test numbers are invalid, the DEXP writes the following message to the OC file:

•VOID RANGE

4.31.5 Show File Derivative

This command displays the contents of a specified command file to the OC file.

Syntax

DXP>SHOW FILE [,filename] [,range$_1$ [,...,range$_n$]]

FILE indicates that a command file is being shown.

filename specifies the one to eight character name of the command file being shown.

range specifies a single memory address or a range of memory addresses separated by a hyphen if showing an inclusive block of memory.

If no filename argument is specified, this command is equivalent to the Show Directory command.

4.32 SNAP Command

This command writes the contents of a specified range of memory locations to the operator communications (OC) file. The memory locations can be task relative (from the task's first memory location) or zero relative (from the system's first memory location).

Syntax

DXP>SNAP { SPAD / taskname } ,range$_1$ [,...,range$_n$]

taskname specifies a one to eight character name of the program if the address being snapped is task relative.

SPAD indicates the CPU's scratchpad memory is being snapped.

range specifies a single memory address or a range of memory addresses separated by a hyphen if snapping an inclusive block of memory addresses. The scratchpad address range is 00_{16} through FF_{16}. If the SPAD argument is specified, all ranges are optional. However, if the memory addresses being snapped are task relative, at least one range must be specified.

If only one address is specified, the function is identical to an Examine command.

Examples

```
DXP>SNAP IP.FDD.r,708C-709E
```

This command writes the contents of memory between locations $708C_{16}$ and $709E_{16}$ in the Gould IOP Floppy Disc Diagnostic.

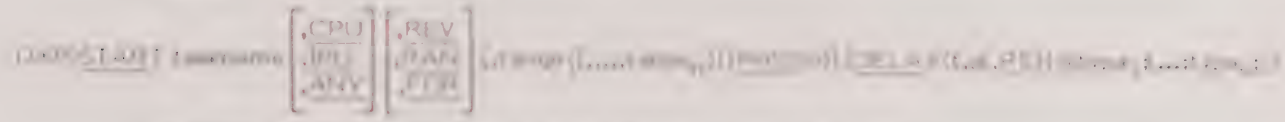
```
DXP>SNAP SPAD 0-FF
```

This command writes the contents of scratchpad between locations 00_{16} and FF_{16}.

4.33 START Command

This command starts program execution and resets its pass counter.

Syntax



taskname specifies the one to eight character name of the program being started.

CPU indicates that the task runs on the central processing unit only.

IPU indicates that the task runs on the internal processing unit only.

ANY indicates that the task runs on either the CPU or IPU as available.

REV indicates that the selected tests run in reverse order.

RAN indicates that the selected tests run in random order.

FOR indicates that the selected tests run in forward order.

range specifies an individual test number or a range of test numbers separated by a hyphen if executing an inclusive block of tests. END can be specified instead of the last test number as the second parameter of the range. If no range is specified, the program executes all defined tests.

PASS(n) indicates the number of passes the program runs.

DELAY indicates performing a delay between tests and/or passes as specified by the subfields.

- t specifies, in seconds, the inter-test delay interval beginning upon completion of each test.
- p specifies, in seconds, the inter-pass delay interval beginning upon completion of each pass.

  Note: The subfields in the Delay argument are order dependant and therefore, zero (null value) or some valid time value must be specified for each delay interval (inter-test and inter-pass) parameter.
- P indicates the subfield entry specifying that the preceding delay intervals are intra-test/pass intervals.

@ specifies that the following is type-ahead data.

typa specifies the type-ahead data. Delimit individual type-ahead entries with a semicolon.

Note: If reverse (REV) or random (RAN) order is specified in the Start command of a program which does not support these functions, the program runs in forward sequential order and writes the following message to the OC file:

```
*TASK DOESN'T ALLOW NON-SEQUENTIAL TESTING
```

Once started, a program may be restarted using the Restart command.

Examples

```
DXP>STAR IP.FDD.r
```

This command starts the Gould IOP Floppy Disc Diagnostic and executes all tests.

```
DXP>STAR IP.FDD.r,3-7,14-17
```

This command starts the Gould IOP Floppy Disc Diagnostic and executes only Tests 3 through 7, and 14 through 17.

```
DXP>STAR IP.FDD.r,CPU,REV,PASS(10)
```

This command starts the Gould IOP Floppy Disc Diagnostic. The program runs 10 passes on the CPU in reverse order.

```
DXP>STAR IP.FDD.r,IPU,DELAY(10,,P)@C;
```

This command starts the Gould IOP Floppy Disc Diagnostic. The task runs on the IPU only with an intra-test delay interval of 10 seconds and automatically responds to the first prompt with C to continue program execution.

4.34 STATUS Command

This command writes the status of a program, the number of DEXP spurious interrupts, or controls the automatic status processor.

4.34.1 Automatic Status Processor Derivative

This derivative controls the automatic status processor.

Syntax

DXP>STATUS [ON | OFF]

ON — indicates that the system snapshots task status every ten minutes and writes a system performance summary including memory addresses, task execution flags, current status, and CPU/IPU execution times to the listed output (LO) file. If the LO file assignment is to the operator communications (OC) file, the system displays no status.

OFF — indicates that no status snapshot occurs.

Note: Setting DEXP Option 1 (enables status display every 10 minutes) and Option 2 (enables system performance data displays) is similar to setting status ON and setting status ON automatically sets Options 1 and 2.

4.34.2 Spurious Interrupt Derivative

This derivative writes the number of DEXP spurious interrupts to the LO file.

Syntax

DXP>STATUS INTERRUPT

4.34.3 Task Status Derivative

This derivative writes the system performance information including the memory address, task execution flags, and current status of a program.

Syntax

```
DXP>STATUS [taskname]
```

taskname specifies the one to eight character taskname of the program requesting status.

If a Status command is executed, the DEXP writes the following message:

```
'taskname' MEMORY:fffff-lllll,STATE:nnnnnnnn { INACTIVE
                                             { RUNNING TEST:n
                                             { DYNAMICALLY SUSPENDED
```

taskname	specifies the one to eight character name of the task.
fffff-lllll	specifies the first memory address (fffff) and last memory address (lllll).
nnnnnnnn	specifies the contents of the task execution status word which reflects the current condition of the task.
INACTIVE	indicates that the task is currently inactive (not executing).
RUNNING	indicates that the task is currently active (executing). If the task is a program, 'n' specifies which test is executing.
DYNAMICALLY SUSPENDED	indicates that the task is waiting for I/O in progress.

If the task is an I/O handler or program, the DEXP writes the following message to the OC file:

```
CHANNEL INTERFACES:n,
INPUT BUFFER:iiiiii, OUTPUT BUFFER:oooooo
CHANNEL ADDR:ca, DCB-ADDR:ddddd, CLASS:c, PRI-LEV:pp,TYPE CODE:tt
DEVICE ADDR(S):#j
```

n	specifies the number of channel interfaces (normally one) that the I/O handler/program supports simultaneously.
iiiiii	specifies the input buffer address.
oooooo	specifies the output buffer address.
ca	specifies the channel address.
ddddd	specifies the device control block (DCB) address for channel address.
c	specifies the I/O class (3, E, or F) of the channel address.
pp	specifies the priority level associated with the channel address.
tt	specifies the assignment device type code associated with the channel address.
j	specifies a hexadecimal number 0 through F.

4.35 TEST Command

This command checks the current state of program options or tests bits at specified memory locations and writes the following message:

```
{OPT}  {##:}  {SET}
{BIT}  {   }  {RES}
```

4.35.1 Option Word Derivative

Syntax

DXP>TEST [taskname],OPT,opt$_1$ [,...,opt$_n$]

taskname — specifies the one to eight character name of the program whose options are being tested. If no taskname is specified, this command tests the Diagnostic Executive Program (CN.DXP.r) options.

OPT — indicates that options are being tested.

opt — specifies the options being tested.

Example

```
DXP>TEST IP.FDD.r,OPT,8
```

This command tests the option word of the Gould IOP Floppy Disc Diagnostic to determine the set/reset condition of Options 8 and writes the following message:

```
OPT 8: SET
```

4.35.2 Bit Manipulation Derivative

Syntax

DXP>TEST [taskname],address,BIT,bit$_1$ [,...,bit$_n$]

taskname — specifies the one to eight character name of the program whose bits are being tested. If no taskname is specified, this command tests specified bits in memory.

address — specifies the hexadecimal address of the word beinge tested.

BIT — indicates that bits are being tested.

bit — specifies the bits being tested.

Example

```
DXP>TEST IP.FDD.r,780C,BIT,1,4
```

This command tests bits 1 and 4 in memory location $780C_{16}$ of the Gould IOP Floppy Disc Diagnostic.

TIME
UPSPACE
WAIT

4.36 TIME Command

This command writes the system elapsed time to the operator communications (OC) file.

Syntax

```
DXP>TIME
```

4.37 UPSPACE COMMAND

This command formats output by writing a specified number of blank lines (spaces) to an output file.

Syntax

```
DXP>UPSPACE lfc [,ct]
```

lfc specifies the two-character code of the file being upspaced (either OC or LO).

ct specifies the number of lines to be upspaced. If the count is not specified, the command upspaces the file by one line.

Example

```
DXP>UPSP LO,5
```

This command upspaces the listed output (LO) file by five lines.

4.38 WAIT Command

This command causes suspension of command file processing if in Batch mode, or places Interactive mode in a wait state (removing it from concurrent task execution).

Syntax

```
DXP>WAIT
```

To exit the wait state, enter @@A.

CHAPTER 5

DIRECTIVE LANGUAGE

5.1 Directive Language Overview

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. Create new tests or reorganize existing tests using directive language.

These operator-composed tests help to examine hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

5.2 Creating Directive Tests

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

Syntax

DXP>taskname $dirtv_1$ [,...,$dirtv_n$]

dirtv specifies a directive.

The following rules apply when creating directive tests:

- Delimit all directives by commas.
- The program ignores spaces which may be inserted for readability.
- Directive order is not important except where illogical.
- Terminate a directive string by pressing the Return key not preceded by a comma.
- Enter the taskname only on the first line of a multiple line directive string. To continue a directive string on the next line, terminate the current line with a comma, press the Return key, and enter the remaining directives in response to the DXP> prompt.

5.3 Editing Directive Tests

If an error is detected prior to pressing the Return key, eliminate an erroneous character by entering a backspace character, a Control-H, or a backspace arrow and reentering the desired character. Perform multiple deletions by successive backspacing.

Erase an entire directive string by entering a delete (DEL) or a rub-out (RUB) character. The DEXP responds to a delete or a rub-out with a carriage return, a line feed, and a new prompt.

5.4 Using Directives

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, the DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP> taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. The DEXP then performs the directive functions and continues execution.

In Batch mode

DXP> BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP> taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to Chapter 4.)

5.5 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Syntax

```
dirtv= ' kw1(sf1,...,sfn),...,kwn '
```

kw specifies a keyword.

sf specifies a subfield.

These arguments contain either a keyword, or a keyword with an associated subfield. A subfield modifies a keyword and if a subfield is present, the keyword/subfield combination is an argument. Delineate subfields with parentheses.

Example

```
RR='OC,SIL,BC(128)'
```

This directive RR inserts a Read I/O command doubleword (IOCD) in the I/O Command List (IOCL) queue, command chained (OC) with suppress incorrect length indication (SIL), and a byte count of 128 (BC(128)).

5.6 Directive Macros

The Diagnostic Executive Program provides the capability of building and executing directive strings defined as macros in all level two macrodiagnostic programs. These macros are either permanently coded (assembled into the program) or temporarily coded (created and defined after program assembly).

Create temporary directive macros with the DEXP's Define command and delete them with the Delete command. (Permanent macros cannot be deleted or redefined).

For more information on creating temporary macros, refer to the Define and Delete commands in Chapter 4.

5.7 Basic Directive Types

There are four basic types of directives:

- Simple directives which contain only alphabetic and numeric components
- Literal directives which contain quoted values within a literal string
- Parameter directives which manipulate entries in the program's parameter table
- Jump directives which transfer control within a directive string

5.7.1 Simple Directives

Simple directives contain only alphabetic and numeric components. For example, in directive CR15, the alphabetic component CR is an acronym for Cycle Output Buffer Right. The optional numeric component 15 specifies the quantities: size, time, or number, in this case 15 byte positions.

Specify the numeric component in decimal (15) or hexadecimal by preceding the number with a single quote ('F). If a simple directive has no modifier, it executes its default condition.

5.7.2 Literal Directives

Literal directives contain an equal sign followed by values enclosed in quotes or parentheses. Use literal directives to load specific values into a specified program location, to enter legal arguments, or to extend or modify the function of a directive.

Syntax

dirtv='arg_1,...,arg_n' or dirtv(arg_1,...,arg_n)

Example

L='ABCD'

This directive loads the characters ABCD in the program's output buffer.

5.7.3 Parameter Directives

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Syntax

$P_{p1}(op)\ \begin{Bmatrix} k \\ p2 \end{Bmatrix}$

All parameter directives start with the letter P and contain the following components:

p_1 specifies an alphabetic character A through Z.

op specifies one of the following valid operand values:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

k specifies a constant.

p_2 specifies a second parameter identifier.

Example

PB*L

If parameter B contains a value of 50 and parameter L contains a value of 2, then PB*L multiplies the value in parameter B (50) by the value in parameter L (2) and stores the result (100) in parameter B. If another PB*L executes now that the value in parameter B is 100, the mathematical equivalent is 100 x 2 = 200, with parameter B storing the result.

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

5.7.4 Jump Directives

Jump directives transfer control within a directive string. Create tests with statement numbers to indicate that the directive immediately following the statement number is the target of the jump.

Example

```
4,LBR,...,J4
```

Four is the statement number that defines the target for the subsequent jump. J4 is the directive LBR.

Conditional Jumps

Conditional jumps specify the contents of a parameter as the iteration count or use a mathematical relationship as an iteration count.

Example

```
P1=9,3,.....J3/L
```

Parameter L contains the value nine. The jump directive causes a transfer of control to the directives associated with statement number three for the number of times specified by the contents of parameter L, in this case nine.

"Computed Go To" Jumps

Use jump directives as "computed go to" jumps to reference statement numbers in a directive string.

5.8 Standard Directives

The Diagnostic Executive Program contains nine categories of standard directives which are available to all level two macrodiagnostic programs. In addition, individual programs implement unique directives which perform functions specific to the program.

The following sections contain a list of the standard directives and a summary of their functions.

5.8.1 "A" Directives (Assign)

Three "A" directives assign channel, controller, and unit addresses for each controller.

A

Uses the channel address and the device subaddress currently in the device definition table (DDT) to create Command Device (CD) and Test Device (TD) instructions.

ACAn

Uses the channel address specified by (n) or, if (n) is not specified, the channel address currently in the DDT to create CD and TD instructions.

ADSAn
AUn

Uses the device subaddress specified by (n) or, if (n) is not specified, the device subaddress currently in the DDT to create CD and TD instructions.

5.8.2 "C" Directives (Cycle)

Four "C" directives cycle the output buffer (equivalent to a circular shift with end-to-end wraparound).

CLn

Cycles the output buffer left by number (n) byte positions. If number (n) is not specified, the DEXP cycles the output buffer left by one byte position.

CLp

Cycles the output buffer left by the value in p (any valid parameter A through Y). If the value in the specified parameter is zero, the DEXP cycles the output buffer left by one byte position.

CRn

Cycles the output buffer right by number (n) byte positions. If number (n) is not specified, the DEXP cycles the output buffer right by one byte position.

CRp

Cycles the output buffer right by the value in p (any valid parameter A through Y). If the value in the specified parameter is zero, the DEXP cycles the output buffer right by one byte position.

5.8.3 "D" Directives (Dump)

Four "D" directives dump all or any specified part of the program's input buffer, output buffer, or parameter set to the listed output (LO) file. If the LO file is assigned to a line printer, the DEXP rounds the byte count up to the next multiple of 32 bytes (eight words). Otherwise, the DEXP rounds the byte count up to the next multiple of 16 bytes (four words).

DIn

Dumps (n) bytes or, if number (n) is not specified, parameter W bytes of the program's input buffer. If number (n) is not specified and parameter W equals zero, the DEXP dumps the entire input buffer.

DOn

Dumps (n) bytes or, if number (n) is not specified, parameter W bytes of the program's output buffer. If number (n) is not specified and parameter W equals zero, the DEXP dumps the entire output buffer.

DP

Dumps the contents of all parameters (A through Z).

DPy

Dumps only the contents of parameter y (any specified parameter A through Z).

5.8.4 "H" Directives (Hold)

Two "H" directives hold program execution. These directives establish delays between directives.

Hn

Holds program execution for (n) number of time units or, if number (n) is not specified, for the number of time units in parameter X. If parameter X equals zero, the hold duration is one time unit (determined by the Real-Time Clock rate). If the Real-Time Clock rate is 120 Hz, one time unit equals 8.3 ms.

HSn

Holds program execution for (n) number of seconds or, if number (n) is not specified, for the number of seconds in parameter X. If parameter X equals zero, the hold duration is one second.

5.8.5 "J" Directives (Jump)

Six "J" directives transfer control within a directive string. Use the following relational operands with conditional jumps:

(op):		
	-	not
	=	equal to
	>	greater than
	<	less than
	=>	equal to or greater than
	>=	greater than or equal to
	<=	less than or equal to
	=<	equal to or less than

Jn

Jumps indefinitely to the statement number (n) (0 < =n < =9).

JZ

Jumps indefinitely to the statement number in parameter Z.

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

Jumps k times (1<k<=32767) to the statement number (n) or the statement number in parameter Z.

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

Jumps to the statement number (n) or the statement number in parameter Z the number of times defined by the contents of the specified (p) parameter (A through Y).

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

Jumps to the statement number (n) or the statement number in parameter Z as long as the contents of p (any specified parameter A through Y) satisfies the stated relationship (op) with the constant (k).

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

Jumps to the statement number (n) or the statement number in parameter Z if the the contents of p_1 (the first specified parameter A through Y) satisifies the stated relationship (op) with the contents of p_2 (the second specified parameter A through Y).

5.8.6 "L" Directives (Load)

Fifteen "L" directives load input or output buffers with various data test patterns.

L=' '

Loads the output buffer with 'any literal value' and stores the literal string length in parameter W.

LAn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with the 64-character printable ASCII subset. If parameter W equals zero, the DEXP loads the entire output buffer.

LBKn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with ASCII blanks (20,20,20,...,20,20,20,...). If parameter W equals zero, the DEXP loads the entire output buffer.

LBPn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with a binary progression (00,01,02,...,FE,FF,00,...). If parameter W equals zero, the DEXP loads the entire output buffer.

LBRn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with a binary regression (FF,FE,FD,...,01,00,FF,...). If parameter W equals zero, the DEXP loads the entire output buffer.

LFOn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with a floating one (01,02,04,...,40,80,01,...) pattern. If parameter W equals zero, the DEXP loads the entire output buffer.

LFZn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with a floating zero (FE,FD,FB,...,BF,7F,FE,...) pattern. If parameter W equals zero, the DEXP loads the entire output buffer.

LIOn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the input buffer with the contents of the output buffer. If parameter W equals zero, the DEXP loads the entire input buffer.

LIRn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with the initial random number pattern. If parameter W equals zero, the DEXP loads the entire output buffer.

LIZn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the input buffer with all zeros. If parameter W equals zero, the DEXP loads the entire input buffer.

LLRn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with the last random number pattern. If parameter W equals zero, the DEXP loads the entire output buffer.

LNRn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with the next random number pattern. If parameter W equals zero, the DEXP loads the entire output buffer.

LOn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with all ones. If parameter W equals zero, the DEXP loads the entire output buffer.

LOIn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with the contents of the input buffer. If parameter W equals zero, the DEXP loads the entire output buffer.

LZn

Loads (n) bytes or, if number (n) is not specified, loads parameter W bytes of the output buffer with all zeros. If parameter W equals zero, the DEXP loads the entire output buffer.

5.8.7 "P" Directives (Parameter)

Five "P" directives manipulate entries in the program's parameter table. Use the following arithmetic and logical operands with "P" directives:

(op):		
	=	replacement
	+	addition
	-	subtraction
	*	multiplication
	/	division
	.A.	logical AND
	.O.	logical OR
	.E.	logical EXCLUSIVE OR
	.N.	two's complement
	.C.	one's complement
	>	circular shift right
	<	circular shift left

Pp(op)k

The DEXP obtains the contents of the specified parameter (p specifies a parameter A through Z), performs the logical/arithmetic operation, and stores the result in the specified parameter (the range for k is zero through 32,767).

$Pp_1(op)p_2$

The DEXP obtains the contents of the specified parameters p_1 and p_2, performs the logical/arithmetic operation, and stores the result in the specified parameter p_1.

PD(op) $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$

The DEXP obtains the contents of parameter D and the second parameter (p) or constant (k), performs the logical/arithmetic operation, and stores the result in parameter D and the output buffer.

PI(op) $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$

The DEXP obtains the contents of parameter I and either the second parameter (Pp) (if specified) or a constant (k), performs the logical/arithmetic operation, and stores the result in parameter I and in Word 1 of the output buffer.

PZ= $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$

The DEXP stores the contents of Pp (p specifies any parameter A through Y) or a constant (k) in parameter Z. The value of parameter Z represents a statement number to which the JZ directive transfers control. The only valid operator for this directive is = (replacement). The values of the specified parameter or constant must be between zero and nine.

5.8.8 "S" Directive (Stop)

The "S" directive creates suspensions in a directive string for performing functions.

STPn

Stops program execution and issues the following prompt to the operator communications (OC) file:

```
'taskname' ,STPn, DIRECTIVE ; [C,H,T] ?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

5.8.9 "T" Directive (Time)

The "T" directive writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

TIME
TIME (message)

Gould CONCEPT/32™

Interval Timer

(CNLITD.r)

Diagnostic Description

Diagnostic Number: 111032-290r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005690-000

™CONCEPT/32 is a Trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Interval Timer (CNLITD.r) Diagnostic Description Release 1.0, Publication Order Number 326-005690-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-4
3-1/3-2
4-1 through 4-2
5-1/5-2
6-1 through 6-8

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 IOP/RTOM Jumper Configurations 2-1
2.3 Program Initialization 2-2
2.4 Program Option Set 2-4

CHAPTER 3 PROGRAM EXECUTION

3.1 Starting Program Execution 3-1
3.2 Terminating Program Execution 3-1

CHAPTER 4 ERROR MESSAGES

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Interrupt Instruction Verification 5-1
5.3 Test 2 Command Verification (Timer Disabled) 5-1
5.4 Test 3 Multiple Interrupt Test 5-1
5.5 Test 4 Multiple Interrupt Test (RTC Frequency) 5-1
5.6 Test 5 Single Interrupt Test 5-1
5.7 Test 6 Single Interrupt Test (RTC Frequency) 5-1

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould CONCEPT/32 Interval Timer Diagnostic Unique Directives 6-7
6.6.1 "C" Directives 6-7
6.6.2 "D" Directive 6-7
6.6.3 "E" Directives 6-7
6.6.4 "F" Directive 6-8
6.6.5 "R" Directives 6-8
6.6.6 "W" Directives 6-8

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould CONCEPT/32 Interval Timer Diagnostic (CN.ITD.r) is a level two macrodiagnostic program which loads and executes under control of the Gould CONCEPT/32 Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of six tests which verify the correct operation of RTCM or IOP Interval Timer functions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode).

1.2 Resource Requirements

The Gould CONCEPT/32 Interval Timer Diagnostic requires:

- A Gould CONCEPT/32 CPU with a minimum of 64K words of memory
- One to four IOPs (Model 8000 or 8001) or
  One to five RTOMs (Model 2345)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

By entering a Return key in response to the prompts, the Gould CONCEPT/32 Interval Timer Diagnostic assigns the following default values during initialization:

- Channel address is 7F
- Default configuration is with an IOP
- Frequency rate is 38.4 microseconds
- Low frequency rate is 38.4 microseconds
- High frequency rate is 2.4 microseconds

To modify the default values, respond to the initialization prompts with one of the responses written with the prompt or use the DEXP Assign command.

Note: Values modified during initialization or by an Assign command become the default values for all reinitialization sequences until reloading and initializing the program.

2.2 IOP/RTOM Jumper Configurations

The Gould CONCEPT/32 Interval Timer Diagnostic uses the Real-Time Clock rate selected during the Diagnostic Executive Program (DEXP) initialization sequence to time the various frequencies available through jumper selection on the IOP or RTOM. The program uses the frequency rate tables written to the operator communications (OC) file during the Gould CONCEPT/32 Interval Timer Diagnostic's initialization sequence to verify or select desired interval timer frequencies and the Real-Time Clock rate.

Note: Failure to define the Real-Time Clock rate correctly causes the program to report errors that may not be indicative of hardware malfunctions.

2.3 Program Initialization

Before initializing the Gould CONCEPT/32 Interval Timer Diagnostic, load it into the system by following the loading procedures in the Diagnostic Facility User's Manual. The program name for loading the Gould CONCEPT/32 Interval Timer Diagnostic is CN.ITD.r. After the program loads, it writes the following initialization sequence to the operator communications (OC) file:

Note: The program reissues any incorrectly answered initialization sequence prompts.

1. The program writes the following message:

```
'taskname' INITIALIZATION
```

2. The program then issues the following prompt:

INTERVAL TIMER CHANNEL ADDRESS=7F; [CR,##]?

Responding with the Return key causes the program to assign the specified default channel address (7F). To modify the default channel address, enter the valid two-character hexadecimal channel address.

The range of valid interval timer channel addresses is 00_{16} through $7F_{16}$.

The program then checks the scratchpad entry for that address to ensure a valid device is configured. If the scratchpad entry does not specify a valid device, the program reissues the prompt.

3. The program then issues the following prompt:

INTERVAL TIMER=IOP; [CR,IOP,RTOM]?

Responding with the Return key causes the program to establish parameters for the specified default configuration (IOP). Responding with RTOM causes the program to establish parameters for the RTOM configuration.

4. Depending upon the response entered in Step 3, the program issues one of the following frequency rate prompts:

4A. If IOP is selected in response to the prompt in Step 3, the program then issues the following prompt:

SELECTED FREQUENCY RATE=38.4; [CR,?,####]?

Responding with the Return key causes the program to select the specified default frequency rate (38.4 microseconds). To modify the default frequency rate, enter any valid frequency rate (in microseconds) in response to the prompt. Responding with a question mark (?) causes the program to write the following table listing all valid frequency rates and corresponding jumper configurations and reissue the prompt:

RATES	IOP MODEL 8000
.6 US	C14.4 - C14.13
1.2 US	C14.3 - C14.14
2.4 US	C14.2 - C14.15
4.8 US	C14.1 - C14.16
9.6 US	C14.8 - C14.9
19.2 US	C14.7 - C14.10
38.4 US	C14.6 - C14.11
76.8 US	C14.5 - C14.12
(60HZ)	E16.7 - E16.10
(120HZ)	E16.6 - E16.11

After the program reissues the prompt, enter one of the frequency rates.

Note: The rates specified in parentheses, (60HZ) and (120HZ), are not valid responses to the frequency rate prompt. They are specified for jumper configuration information only.

4B. If an RTOM is selected in response to the prompt in Step 3, the program issues the following prompts:

```
SELECTED LOW FREQUENCY RATE = 38.4; [CR,?, ####]?
```

Responding with the Return key causes the program to select the specified default low frequency rate (38.4 microseconds). To modify the default low frequency rate, enter any valid low frequency rate (in microseconds) in response to the prompt. Responding with a question mark (?) causes the program to write the following table which lists all valid low frequency rates and corresponding jumper configurations and reissue the prompt:

RATES	RTOM MODEL 2345
4.8 US	U158.4 - U158.13
9.6 US	U158.3 - U158.14
19.2 US	U158.2 - U158.15
38.4 US	U158.1 - U158.16
(60HZ)	U107.1 - U107.16
(120HZ)	U107.2 - U107.15

After the program reissues the prompt, enter one of the frequency rates.

Note: The rates specified in parentheses, (60HZ) and (120HZ), are not valid responses to the low frequency rate prompt. They are specified for jumper configuration information only.

The program then issues the following prompt:

```
SELECTED HIGH FREQUENCY RATE=2.4;[CR,?,###]?
```

Responding with the Return key causes the program to select the specified default high frequency rate (2.4 microseconds). To modify the default high frequency rate, enter any valid high frequency rate (in microseconds) in response to the prompt. Responding with a question mark (?) causes the program to write the following table which lists all valid high frequency rates and corresponding jumper configurations and reissue the prompt:

RATES	RTOM MODEL 2345
.3 US	U158.8 - U158.9
.6 US	U158.7 - U158.10
1.2 US	U158.6 - U158.11
2.4 US	U158.5 - U158.12
(60 HZ)	U107.1 - U107.16
(120 HZ)	U107.2 - U107.15

After the program reissues the prompt, enter one of the frequency rates.

Note: The rates specified in parentheses, (60HZ) and (120HZ), are not valid responses to the high frequency rate prompt. They are specified for jumper configuration information only.

2.4 Program Option Set

Once loaded and initialized, the Gould CONCEPT/32 Interval Timer Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould CONCEPT/32 Interval Timer Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Unimplemented.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file upon completion of each pass.
11	Set	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.

CHAPTER 3

PROGRAM EXECUTION

3.1 Starting Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#111032-290r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR MESSAGES

4.1 Standard Error Messages

The Gould CONCEPT/32 Interval Timer Diagnostic (CN.ITD.r) writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<<'taskname'----hh:mm:ss>>>>
   ```

taskname	specifies the one to eight character program name as indicated in the load map.
hh:mm:ss	specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TST:tt, DEVICE:dd, PC:aaaaa
   ```

tt	specifies the decimal test number of the erring test.
dd	specifies the hexadecimal device address assigned.
aaaaa	specifies the hexadecimal program counter value within the erring test.

3. The erring directive (for directive tests only)

   ```
   DRTV:directive
   ```

directive	specifies the erring directive.

4. The I/O command issued

   ```
   CMND:dddddddd(ccc), COUNT:nnnnnnnn,  FREQ:ffff
   ```

dddddddd	specifies the last interval timer instruction executed (in hexadecimal).
ccc	specifies the command mnemonic.
nnnnnnnn	specifies the initial value of the interval timer's register count.
ffff	specifies the current frequency under test.

4.2 Unique Error Messages

If the interval timer's register is read and a value other than the expected value is encountered, the program writes the following message:

```
ERROR READING TIMER'S REGISTER
ACT dddddddd
EXP dddddddd
```

In the previous message, the program issues a CHKC directive but the input buffer value does not equal the pre-established test value in parameter C.

If a command to write the interval timer's register results in changing the contents of general purpose register zero (GPR0), the program writes the following message:

```
ERROR WRITING TIMER'S REGISTER
ACT dddddddd
EXP dddddddd
```

In the previous message, the expected value is the original GPR0 contents and the actual value is the final GPR0 contents after the Write command.

If the actual number of service interrupts received from the interval timer does not fall within the given range, the program writes the following message:

```
iiii INTERRUPTS RECEIVED; xxxx-yyyy EXPECTED
```

iiii — specifies the actual number of interrupts received during the allotted time (100 ms).

xxxx — specifies the lowest acceptable number of interrupts to receive for the frequency under test.

yyyy — specifies the highest acceptable number of interrupts to receive for the frequency under test.

If in Test 1, an interrupt control instruction is executed in which a service interrupt (SI) is expected but is not received, the program writes the following message:

```
NO SI
```

If in Test 1, an interrupt control instruction is issued in which no service interrupt (SI) is expected but one is received, the program writes the following message:

```
UNEXPECTED SI
```

If in Tests 4 and 6, the interval timer's register fails to continue counting negative after generating a service interrupt in response to a single interrupt command, the program writes the following message:

```
COUNT FAILED TO GO NEGATIVE
```

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould CONCEPT/32 Interval Timer Diagnostic (CN.ITD.r) contains a sequence of six tests which run either in a continuous sequence or individually by operator request. The tests use a 38.4 microsecond low frequency rate and a 2.4 microsecond high frequency rate as default values on a RTDM, and a 38.4 microsecond default rate for an IOP.

Execute Tests 3 through 6 only with Clock Override OFF since these tests require interrupts from the interval timer. If the Clock Override function is ON, it inhibits all interrupts from the interval timer.

5.2 Test 1 Interrupt Instruction Verification

This test verifies that the following basic interrupt control instructions: Disable (DI), Enable (EI), Request (RI), Activate (AI), and Deactivate (DAI) perform as specified.

5.3 Test 2 Command Verification (Timer Disabled)

This test performs basic read and write functions to the interval timer with the counter disabled. The program tests all bits of the counting register to ensure that each is turned ON and OFF and that no cross talk exists between the bits.

5.4 Test 3 Multiple Interrupt Test

This test verifies that all combinations of Command Device (CD) instructions with multiple interrupt functions set place the interval timer in the multiple interrupt mode. If a RTDM is configured, the program repeats the test for the high frequency rate.

5.5 Test 4 Multiple Interrupt Test (RTC Frequency)

This test performs the same function as Test 3 using the Real-Time clock frequency.

5.6 Test 5 Single Interrupt Test

This test verifies that all combinations of Command Device (CD) instructions with single interrupt functions (multiple reset) place the interval timer in the single interrupt mode. If a RTDM is configured, the program repeats the test for the high frequency rate.

5.7 Test 6 Single Interrupt Test (RTC Frequency)

This test performs the same functions as Test 5 using the Real-Time Clock frequency.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould CONCEPT/32 Interval Timer Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

in a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould CONCEPT/32 Interval Timer Diagnostic parameters and a summary of their functions follow:

Parameter	Type	Description
A	Reserved	Contains the input/output buffer adddress attributes (F=0,C=00).
B	Available	Available for general use.
C	Reserved	Contains the comparison value for counter reads.
D	Reserved	Contains the output buffer which the program uses to load the count into the interval timer register.
E	Available	Available for general use.
F	Reserved	Contains the interval timer frequency control bits.
G	Available	Available for general use.
H	Reserved	Contains the high limit for interrupt comparison.
I	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Reserved	Contains the low limit for interrupt comparison.
M	Reserved	Contains the device status mask.
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Reserved	Contains the total number of interrupts received from a directive test.
R	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Available	Available for general use.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for JZ directives.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L='' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

PI(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

PZ= $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould CONCEPT/32 Interval Timer Diagnostic Unique Directives

The Gould CONCEPT/32 Interval Timer Diagnostic contains unique directives which perform functions specific to the program.

The Gould CONCEPT/32 Interval Timer Diagnostic unique directives follow.

6.6.1 "C" Directives

CHKC

Tests the contents of the input buffer against parameter C. If not equal, the directive writes an error message.

CHKI

Checks that the number of interrupts received (parameter I) falls within the limits set by parameter L and parameter H.

6.6.2 "D" Directive

DT

Disables the interval timer.

6.6.3 "E" Directives

EMI

Enables the interval timer with multiple interrupt functions.

ESI

Enables the interval timer with single interrupt functions.

6.6.4 "F" Directive

FRQ='nnnn'

Selects which frequency to test and determines the acceptable interrupt limits for that frequency. The frequency is changed if the following conditions exist:

nnnn=		
.3	selects 300 nanosecond frequency	High Frequency*, RTOM only
.6	selects 600 microsecond frequency	IOP and RTOM High Frequency*
1.2	selects 1.2 microsecond frequency	IOP and RTOM High Frequency*
2.4	selects 2.4 microsecond frequency	IOP and RTOM High Frequency*
4.8	selects 4.8 microsecond frequency	IOP and RTOM Low Frequency*
9.6	selects 9.6 microsecond frequency	IOP and RTOM Low Frequency*
19.2	selects 19.2 microsecond frequency	IOP and RTOM Low Frequency*
38.4	selects 38.4 microsecond freqeuncy	IOP and RTOM Low Frequency*
76.8	selects 76.8 microsecond frequency	IOP only

* High and Low Frequency apply to RTOM only.

Real-Time Clock frequency is changed if the following conditions exist:

nnnn=	
50	selects a frequency of 20.0 milliseconds (50Hz)
60	selects a frequency of 16.67 milliseconds (60 Hz)
100	selects a frequency of 10.0 milliseconds (100 Hz)
120	selects a frequency of 8.33 milliseconds (120 Hz)
EXT	selects a frequency determined by the external clock

6.6.5 "R" Directives

RD

Reads the interval timer with all other functions disabled.

RLD

Reads and loads the interval timer with the timer disabled.

RLM

Reads and loads the interval timer with multiple interrupt function enabled.

RLS

Reads and loads the interval timer with single interrupt function enabled.

RM

Reads the interval timer with multiple interrupt function enabled.

RS

Reads the interval timer with single interrupt function enabled.

6.6.6 "W" Directives

WDI

Writes the interval timer with all other functions disabled.

WMI

Writes the interval timer with multiple interrupt function enabled.

WSI

Writes the interval timer with single interrupt function enabled.

Gould CONCEPT/32™

Multiport Memory

(CNLMPM.r)

Diagnostic Description

Diagnostic Number: 111032-251r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005680-000

™CONCEPT/32 is a Trademark of Gould Inc.

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT/32 Multiport Memory (CN.MPM.r) Diagnostic Description Release 1.0, Publication Order Number 326-005680-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-5/2-6
3-1 through 3-2
4-1 through 4-2
5-1 through 5-4
6-1 through 6-7/6-8

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Private and Shared Memory Configuration Requirements 2-1
2.3 Program Initialization 2-2
2.4 Program Option Set 2-4

CHAPTER 3 PROGRAM EXECUTION

3.1 Starting Program Execution 3-1
3.2 Single CPU Configuration Execution 3-1
3.3 Multiple CPU Configuration Execution 3-2
3.4 Terminating Program Execution 3-2

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Testing Procedure 5-1
5.2 Testing Modes 5-1
5.3 Tests 1, 2, 3 Fixed Pattern Tests 5-1
5.4 Test 4 Address in Address Test 5-2
5.5 Test 5 Address Complement in Address 5-2
5.6 Test 6 F and C Bit (Byte) 5-2
5.7 Test 7 F and C Bit (Halfword) 5-3
5.8 Test 8 Memory Execution Test 5-3
5.9 Test 9 Memory Contention Test 5-3
5.10 Test 10 Memory Execution and Contention Test 5-4
5.11 Test 11 Moving Inversions 5-4

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould CONCEPT/32 Multiport Memory Diagnostic Unique Directives 6-7
6.6.1 "L" Directives 6-7
6.6.2 "N" Directives 6-7
6.6.3 "T" Directive 6-7

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould CONCEPT/32 Multiport Memory Diagnostic (CN.MPM.r) is a level two macrodiagnostic program that loads and executes under control of the Gould CONCEPT/32 Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 11 tests which verify the correct operation of private or shared memory configured on a CONCEPT/32 CPU. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode).

1.2 Resource Requirements

The Gould CONCEPT/32 Multiport Memory Diagnostic requires:

- A Gould CONCEPT/32 CPU with a minimum of 64K words of memory
- At least one 32 SERIES Memory Bus Adapter for each shared memory port connected to the CONCEPT/32 computer system

Shared memory consists of memory chassis, memory modules, memory bus controller, and X-bus cable. The sum total of private and shared memory may be up to 16 megabytes.

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

By entering the Return key in response to the prompts, the Gould CONCEPT/32 Multiport Memory Diagnostic assigns the following default values during initialization:

- The memory test type is for private memory
- All addresses are computed by the program and are included in the initialization prompts as default values
- If shared memory is indicated, the default number of processors configured is two

To modify the default values, respond to the prompts with one of the responses written with the prompt or use the DEXP Assign command.

Note: Values modified during initialization or by an Assign command become the default values for all reinitialization sequences until reloading and initializing the program.

2.2 Private and Shared Memory, Configuration Requirements

The Gould CONCEPT/32 Multiport Memory Diagnostic (CNMPM.r) tests single CPU configurations with only private memory or multiple CPU configurations with private and shared memory.

Single CPU configurations require one copy of the program. Multiple CPU configurations with shared memory require loading one copy of the program into each CPU.

In multiple CPU configurations with shared memory, first load the program and verify each CPU's private memory before starting the shared memory tests. This reduces contention with other configured CPUs. If switching between private memory testing and shared memory testing, it is not necessary to reload the program, use the DEXP Reinitialize command.

CAUTION

This program should be the last one loaded if multiple macrodiagnostic programs are being executed simultaneously.

2.3 Program Initialization

Before initializing the Gould CONCEPT/32 Multiport Memory Diagnostic, load it into the system by following the loading procedures in the Diagnostic Facility User's Manual. The program name for loading the Gould CONCEPT/32 Multiport Memory Diagnostic is CN.MPM.r. After the program loads, it writes the following initialization sequence to the operator communications (OC) file:

Note: The program reissues any incorrectly answered initialization sequence prompts.

1. The program writes the following message:

```
'taskname' INITIALIZATION
```

2. The program then issues the following prompt:

```
MEMORY TEST TYPE=PRIVATE;[CR,SHARED]?
```

Responding with the Return key causes the program to assign the specified default memory test type (PRIVATE). To modify the default memory test type, enter SHARED for shared memory testing.

3A. If private memory testing is indicated, the program then issues the following prompt:

```
START ADDRESS=ffffff;[CR,######]?
```

The program computes the default address specified (ffffff) to be the first location of the next 8K word boundary following the end of the program. Responding with the Return key causes the program to assign the precalculated default start address as the first location to test. To modify the default start address, enter any valid address within 16 megabytes (######). The specified address must be on a word boundary and after the program.

3B. The program then issues the following prompt:

```
STOP ADDRESS=ssssss;[CR,######]?
```

The program computes the default address specified (ssssss) to be the last location of the 8K word increment containing the start address. Responding with the Return key causes the program to assign the default stop address as the last location to test. To modify the default stop address, enter any valid address within 16 megabytes (######). The address specified must be on a word boundary and equal to or greater than the start address.

Memory between the start address and the stop address must be contiguous with no holes. (For systems with holes in memory, the individual areas must be tested one at a time.)

4A. If shared memory testing is indicated, the program issues the following prompt:

```
PROCESSORS CONFIGURED=2;[CR,#]?
```

Responding with the Return key causes the program to assign the specified default number of processors configured (2). To modify the default number of processors configured, enter the valid decimal number between 3 and 16.

4B. The program then issues the following prompt:

```
PROCESSOR NUMBER=P;[CR,P,S##]?
```

Responding with the Return key causes the program to assign the specified default processor number (P). To modify the default processor number, enter the valid secondary processor number 1 through 15.

4C. The program then issues the following prompt:

```
COMMON ADDRESS=cccccc;[CR,######]?
```

The program computes the default address specified (cccccc) to be the first location of the next 8K word boundary following the end of the program. Since the program can not determine the starting address of shared memory, rarely is the default common address correct. Therefore, enter the known address corresponding to the first location within shared memory. The specified address must be on a word boundary and equal to or greater than the specified default address.

The specified common address must reside within shared memory and represents the beginning address of a 21-word block of memory which contains program flags and parameters the primary processor uses to control and monitor the execution of secondary processors.

4D. The program then issues the following prompt:

```
START ADDRESS=ffffff;[CR,######]?
```

The program computes the default address specified (ffffff) to be the first location following the 21-word common area. Responding with the Return key causes the program to assign the default start address as the first location to test. To modify the default first location to test, enter any address within the shared memory.

4E. The program then issues the following prompt:

```
STOP ADDRESS=ssssss ;[CR,######]?
```

The program computes the default address specified 'ssssss' to be the last location of the 8K word increment containing the start address. Responding with the Return key causes the program to assign the precalculated default stop address as the last location to test. To modify the default stop address, enter any valid address within shared memory. The specified address must be on a word boundary and equal to or greater than the start address.

The memory between the start address and the stop address must be contiguous with no holes.

5. The program then writes the following message for every processor initialized:

```
'taskname' INITIALIZATION COMPLETE
```

The message indicates that initialization is successful. At this time, modify options and/or start the program.

The initialization sequence for a multiple CPU configuration must begin by first initializing the primary (P) processor and then initializing the secondary (S_1 through S_n) processors. If an error occurs while initializing a secondary processor prior to the "Initialization Complete" message for that processor, use the DEXP Abort command to stop the initialization sequence for that processor only. Then use the DEXP Reinitialize command to restart the initialization sequence. Failure to initialize all processors correctly may result in test sequencing errors which are not indicative of hardware malfunctions.

2.4 Program Option Set

Once loaded and initialized, the Gould CONCEPT/32 Multiport Memory Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould CONCEPT/32 Multiport Memory Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.

Option	Default	Effect If Set
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR... ; [C,H,T] ? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file upon completion of every pass.
11	Set	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Inhibits data error messages.
16	Reset	Writes all data error messages.

CHAPTER 3

PROGRAM EXECUTION

3.1 Starting Program Execution

The following information is necessary to ensure the correct start and execution of this program:

- Do not specify RAN (random) order in the Diagnostic Executive Program (DEXP) Start or Restart commands if this program is in a multiple CPU configuration or with multiple copies. RAN order may be specified in a single CPU configuration with a single copy.

- On a CONCEPT 32/67 CPU, the memory range specified in the initialization sequence is not executed from cache/shadow memory. This avoids processor conflicts. Any new memory range specified during reinitialization (without halting and resetting the system) is added to the existing memory range inhibited from executing from cache/shadow memory.

- To enable cache/shadow memory execution of any memory range currently inhibited, the system must be halted and reset. Then to continue execution, the program must be reinitialized following the halt and reset functions.

After setting options, issue a DEXP Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#111032-251r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED

CPU TYPE = 32/xx
```

xx specifies the CONCEPT/32 machine type.

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Single CPU Configuration Execution

Single CPU execution with private memory testing requires no special considerations.

3.3 Multiple CPU Configuration Execution

In multiple CPU configurations, the primary processor controls the execution of the secondary CPUs by manipulating various control flags and monitoring status flags. The primary processor also writes background data patterns for the contention tests.

The secondary processors write and read memory cells during Tests 6, 7, and 8 as well as reading background data patterns for the contention tests.

The start logic within the program requires that all secondary processors be started prior to starting the primary processor. The primary processor then schedules the execution of the secondary processors.

If a range of test numbers is specified during the DEXP Start or Restart command, the same range must be specified for all processors.

3.4 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

CAUTION

Using the Hold command on a secondary processor while executing in the shared memory mode may cause the primary processor to write the following message to the operator communications (OC) file:

'taskname' TERMINATED: TST:ttn,PROCESSOR:#TIMED OUT

The primary processor cannot distinguish between a program on hold from a secondary processor locked out of shared memory due to contention.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP>prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould CONCEPT/32 Multiport Memory Diagnostic (CNMPM.r) writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   <<<<'taskname'----hh:mm:ss>>>>

taskname	specifies the one to eight character program name as indicated in the load map.
hh:mm:ss	specifies the system elapsed execution time at the point of error.

2. The error message header

   TST:ttm, CPU:uuu, (test function)

ttm	specifies the test number (tt) and scheduling mode (m) of the erring test. The scheduling mode (m) is A if scheduling is one processor at a time or B if scheduling is all processors at a time (contention mode).
uuu	specifies the erring processor (P,S1,S2,...Sn).
test function	specifies the particular pattern and test being executed.

3. The erring directive message (for directive tests only)

   DRTV: directive

directive	specifies the erring directive.

4.2 Unique Error Messages

If a data error occurs, the program writes the following message:

```
DATA ERROR(S)
ADDRESS  EXPECTED  ACTUAL
aaaaaa   eeeeeeee  ffffffffffffffff
```

aaaaaa	specifies the word address where the failure occurred.
eeeeeeee	specifies the expected data value.
ffffffff	specifies the actual data value.

If the primary processor times out a secondary processor due to unresolved contention with the memory system, the program writes the following message:

```
'taskname' TERMINATED: TST:ttm, PROCESSOR: # TIMED OUT
```

#	specifies which processor (S1 through S15) times out.

If parity errors occur, the program writes the following message:

```
MEMORY PARITY TRAP ACTUAL PSD pppppppp-pppppppp cccccccc
PARITY FAILURE AT ADDRESS aaaaaaaa
```

pppppppp-pppppppp	specifies the contents of the Program Status Doubleword at the time of error.
cccccccc	specifies the contents of the CPU Status Word at the time of error.
aaaaaaaa	specifies the memory address of the failure.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Testing Procedure

The Gould CONCEPT/32 Multiport Memory Diagnostic (CN.MPM.r) tests the Memory Bus Adapter's ability to communicate data between the SelBUS and remote memory.

In a multiple CPU configuration, issue a DEXP Start command to all secondary processors prior to issuing a Start command for the primary processor. Ensure that all Start command options are the same for all configured processors.

5.2 Testing Modes

There are two execution modes for all tests. The primary processor schedules the mode executed by all secondary processors.

In mode A, all processors sequentially execute tests to avoid contention. This verifies that each processor executes tests without contention when accessing remote memory.

In mode B, all processors concurrently execute tests to create contention between Memory Bus Adapters when accessing remote memory. The primary processor monitors the execution of all secondary processors during mode B and ensures that all processors complete each test without errors in a specified time interval.

Tests 1 through 10 are directive tests. Test 11 is an in-line coded test.

5.3 Tests 1,2,3 Fixed Pattern Tests

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded between the selected start and stop addresses in remote memory of the primary processor with the following fixed patterns:

Test 1	all zeros
Test 2	all ones
Test 3	alternating ones

Memory is tested with these fixed patterns between the start and stop addresses of each configured processor. Then, the scheduling mode is selected and an unconditonal jump is made to the loop point.

5.4 Test 4 Address in Address Test

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded between the selected start and stop addresses of the primary processor with each address being written into its respective word location.

Memory is tested between the start and stop address of each configured processor, then the scheduling mode is selected and an unconditional jump is made to the loop point.

5.5 Test 5 Address Complement in Address

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded between the selected start and stop addresses of the primary processor only, with each address being complemented and written into its respective word location.

Memory is tested between the start and stop address of each configured processor, then the scheduling mode is selected and an unconditional jump is made to the loop point.

5.6 Test 6 F and C Bit (Byte)

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded and tested between the primary processor's start address and the secondary processor's stop address allowing each configured processor to load and test byte resolution concurrently, using an alternating processor identity pattern. The scheduling mode is selected and an unconditional jump is made to the loop point.

5.7 Test 7 F and C Bit (Halfword)

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded and tested between the primary processor's start address and the secondary processor's stop address allowing each configured processor to load and test halfword resolution concurrently, using an alternating processor identity number. The scheduling mode is selected and an unconditional jump is made to the loop point.

5.8 Test 8 Memory Execution Test

The scheduling mode is selected, a loop point is established, and memory is loaded by the primary processor only between its start and stop addresses in 8K increments, with a five-instruction Add Bit in Register program. The test is set to execute by the secondary processors only, allowing the primary processor to "time out" any secondary processors that cannot execute the test properly. The scheduling mode is selected and an unconditional jump is made to the loop point.

If the current processor's start address plus the length of the test program exceeds the 128K word memory boundary, this test is skipped.

5.9 Test 9 Memory Contention Test

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded and tested between each secondary processor's start and stop addresses using of the Set Bit in Memory and Test Bit in Memory instructions where the bit being tested corresponds to the processor's identity number. The scheduling mode is selected and an unconditional jump is made to the loop point.

This test creates contention between Memory Bus Adapters using scheduling mode B. If contention between processors (due to the read and write lock mechanism) cannot be resolved, the primary processor times out the erring secondary processors.

5.10 Test 10 Memory Execution and Contention Test

The scheduling mode is selected, a loop point is established, and the loop on error point is established. Memory is loaded by the primary processor only, with a 28-instruction Set Bit in Memory program, starting at the first address word following the common communication area.

The program is executed by the secondary processors only. Each processor sets bits in memory corresponding its identity number between the secondary processor's start and stop addresses.

The memory is tested between the start and stop addresses of the secondary processors through a Test Bit in Memory instruction where the bit to be tested corresponds to the processor's identity number. The scheduling mode is selected and an unconditional jump is made to the loop point.

This test creates contention between Memory Bus Adapters if executing and setting bits in the shared memory in scheduling mode B. If contention between processors (due to the read and lock mechanism) cannot be resolved, the primary processor times out the erring secondary processors.

If the current processor's start address plus the length of the test program exceeds 128K word memory boundary, the program skips this test.

5.11 Test 11 Moving Inversions

This test is executed during the Gould CONCEPT/32 Multiport Memory Diagnostic's memory test function.

The test issues Read, Write, Read instructions to every location between the start and stop addresses. Addressing is executed in both the forward and reverse directions. Address incrementing also is varied in forward and reverse directions.

The memory is loaded first with zeros from the start and stop addresses. A marching ones pattern is loaded and verified from the start to stop addresses (forward addressing). Then a marching zeros pattern is loaded and verified from the stop to start addresses (reverse addressing).

The address increment is changed from one word to two words. The entire read, write, read (verify, write, verify) sequence (forward/reverse) is repeated. The address increment is updated from two words to four words and again a read, write, read of every address is executed in both directions. The sequence is repeated until the address increment has cycled through all address bits.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The CONCEPT/32 Multiport Memory Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DEXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould CONCEPT/32 Multiport Memory Diagnostic parameters and a summary of their functions follow:

Parameter	Type	Description
A	Reserved	Contains the input/output buffer address attributes (F=0, C=0).
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if set using a PD directive.
E	Available	Available for general use.
F	Available	Available for general use.
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the current output buffer word one data pattern if set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Available	Available for general use.
N	Available	Available for general use.
O	Available	Available for general use.
P	Reserved	Contains the current output buffer word one data pattern if set using a PP directive.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for JZ directives.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L='' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

$PD(op) \begin{Bmatrix} k \\ Pp \end{Bmatrix}$

$PI(op) \begin{Bmatrix} k \\ Pp \end{Bmatrix}$

$PZ= \begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn — Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME
TIME,message — Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould CONCEPT/32 Multiport Memory Diagnostic Unique Directives

The Gould CONCEPT/32 Multiport Memory Diagnostic contains unique directives which perform functions specific to the program.

The Gould CONCEPT/32 Multiport Memory Diagnostic unique directives follow.

6.6.1 "L" Directives

LE

Establishes a loop on error or subtest entry point within a directive sequence.

LMn

Loads the shared memory between the selected start and end addresses (excluding the common area) for the pattern defined by n.

6.6.2 "N" Directives

N(A-Y)

Defines the directive subtest letter identifiers (A through Y).

NZ

Zeros the directive subtest letter identifier.

6.6.3 "T" Directive

TMn

Tests the shared memory between the selected start and end addresses (excluding the common area) for the pattern defined by n.

Gould CONCEPT 32/27

Floating Point

(27.FPT.r)

Diagnostic Description

Diagnostic Number: 271032-270r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005700-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

CONCEPT/32 is a trademark of Gould Inc.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT 32/27 Floating Point (27.FPT.r) Diagnostic Description Release 1.0, Publication Order Number 326-005700-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-2
3-1/3-2
4-1 through 4-2
5-1 through 5-2
6-1 through 6-10

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Option Set 2-1

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Terminating Program Execution 3-1

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Single Precision Add 5-1
5.3 Test 2 Single Precision Subtract 5-1
5.4 Test 3 Single Precision Multiply 5-1
5.5 Test 4 Single Precision Divide 5-2
5.6 Test 5 Double Precision Add 5-2
5.7 Test 6 Double Precision Subtract 5-2
5.8 Test 7 Double Precision Multiply 5-2
5.9 Test 8 Double Precision Divide 5-2
5.10 Test 9 Single Precision Chain 5-2
5.11 Test 10 Double Precision Chain 5-2

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould CONCEPT 32/27 Floating Point Diagnostic Unique Directives 6-7
6.6.1 "A" Directives 6-8
6.6.2 "C" Directives 6-8
6.6.3 "D" Directives 6-9
6.6.4 "M" Directives 6-9
6.6.5 "S" Directives 6-10

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould CONCEPT 32/27 Floating Point Diagnostic (27.FPT.r) is a level two macrodiagnostic program that loads and executes under control of the Gould CONCEPT/32 Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 10 tests which verify the correct operation of a floating point instruction or combination of floating point instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires minimal operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive tests from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode).

1.2 Resource Requirements

The Gould CONCEPT 32/27 Floating Point Diagnostic requires:

- A CONCEPT 32/27 CPU with a minimum of 64K words of memory with firmware level 531-322655-010 or later (for multiwire) or 531-322760-002 (for copper)
- A 32/27 Floating Point Accelerator (optional)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould CONCEPT 32/27 Floating Point Diagnostic is a self-initializing program and sets up its default assignment values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the Diagnostic Facility User's Manual. The program name for the Gould CONCEPT 32/27 Floating Point Diagnostic is 27.FPT.r.

2.2 Program Option Set

Once loaded, the Gould CONCEPT 32/27 Floating Point Diagnostic enters a wait state.

Use the UECP Set or Reset commands to change default option settings, thereby modifying execution conditions.

The Gould CONCEPT 32/27 Floating Point Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T] ? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.

Option	Default	Effect If Set
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Reserved, do not use.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file after every 5000 passes.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Unimplemented.
16	Reset	Simulates error conditions. Setting this option simulates all possible error conditions in program tests, allowing a step-through execution. Use this function as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.
17	Reset	Unimplemented.
18	Reset	Inhibits testing during pass 1 of the program.
19	Reset	Inhibits testing during pass 2 of the program.
20	Reset	Adjusts random operand values.
21	Reset	Enables direct operand addressing mode only.
22	Reset	Unimplemented.
23	Reset	Enables selecting the simulator used to calculate floating point divide results (when used with Option 24).
24	Reset	Selects the simulator to use for calculating the expected floating point division results as follows: Set selects the hardware floating point accelerator simulator, Reset selects the CPU firmware simulator. If Option 23 is reset, the program bypasses this option.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#271032-270r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP>prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould CONCEPT 32/27 Floating Point Diagnostic [27.FPT.r] writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<<'taskname'----hh:mm:ss>>>>
   ```

   taskname — specifies the one to eight character program name as indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TEST:ttt,  PASS:ppppppp,  PC:ccccc
   ```

   ttt — specifies the test and subtest where the error occurred.

   ppppppp — specifies the current pass count.

   ccccc — specifies the hexadecimal memory address of the start of the subtest.

3. The erring directive message (for directive tests only)

   ```
   DRTV:directive
   ```

   directive — specifies the erring directive.

4.2 Unique Error Messages

If an error occurs during execution, the program writes an error message with one or more of the following components:

The erring floating point instruction, the destination register used, and the location of the source operand:

```
iiiii  DST:dd,  SRC:sssss
```

iiiii	specifies the floating point instruction in error.
dd	specifies the destination register of the instruction.
sssss	specifies the hexadecimal memory address of the source operand.

The destination and source operands:

```
DST OP:nn...nn
SRC OP:nn...nn
```

DST OP	indicates the destination operand value.
SRC OP	indicates the source operand value.
nn...nn	specifies the hexadecimal value (8 or 16 digits) of the indicated operand.

The expected and actual arithmetic results:

```
EXP CC'S:bbbb ANS:nn..nn
ACT CC'S:bbbb ANS:nn..nn
```

EXP CC'S	indicates the expected condition code value.
ACT CC'S	indicates the actual condition code value.
ANS	indicates the expected or actual answer.
bbbb	specifies, in binary, the condition code contents.
nn...nn	specifies the hexadecimal value of the indicated result (8 or 16 digits).

If an arithmetic exception trap is detected, the program writes the following message:

```
EXP AE TRAP PSD:pppppppp pppppppp
ACT AE TRAP PSD:pppppppp pppppppp
```

EXP AE TRAP PSD	indicates the expected arithmetic exception trap program status doubleword.
ACT AE TRAP PSD	indicates the actual arithmetic exception trap program status doubleword.
pppppppp pppppppp	specifies the hexadecimal contents of the trap program status doubleword, or if all zeros, indicates no trap.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould CONCEPT 32/27 Floating Point Diagnostic (27.FPT.r) contains 10 tests which verify the correct operation of one of the floating point instructions or, in the case of Tests 9 and 10, a chained sequence of back-to-back floating point instructions.

Pass #	Tests	Operand Type	Inhibiting Option (if set)
1	1-8	tests boundary conditions	18
	9-10	no operations performed	
2	1-10	general purpose	19
3	1-10	pseudo-random	-

The program calculates the expected result of each floating point instruction. During floating point division instruction tests, the program chooses between two software simulators: one to predict the results calculated by the CPU and the other to predict results calculated by the hardware Floating Point Accelerator. The program selects the appropriate simulator on the basis of the status returned in bit 22 by a Read CPU Status (RDSTS) instruction. Set Option 23 (overrides selection of the simulator) and Option 24 (selects the simulator) to override the selection. The program reports the algorithm in use to the listed output (LO) file after making the first selection and at any time the selection changes.

After the execution of each floating point instruction, the program compares the actual hardware result (data, condition codes, and exceptional condition traps) to the software calculated result, and if different, it writes an error message sequence to the LO file.

5.2 Test 1 Single Precision Add

This test contains 10 subtests which verify the correct execution of the Add Floating Point Word (ADFW) instruction.

5.3 Test 2 Single Precision Subtract

This test contains 10 subtests which verify the correct execution of the Subtract Floating Point Word (SUFW) instruction.

5.4 Test 3 Single Precision Multiply

This test contains 10 subtests which verify the correct execution of the Multiply Floating Point Word (MPFW) instruction.

5.5 Test 4 Single Precision Divide

This test contains 10 subtests which verify the correct execution of the Divide Floating Point Word (DVFW) instruction.

5.6 Test 5 Double Precision Add

This test contains 19 subtests which verify the correct execution of the Add Floating Point Doubleword (ADFD) instruction.

5.7 Test 6 Double Precision Subtract

This test contains 19 subtests which verify the correct execution of the Subtract Floating Point Doubleword (SUFD) instruction.

5.8 Test 7 Double Precision Multiply

This test contains 19 subtests which verify the correct execution of Multiply Floating Point Doubleword (MPFD) instruction.

5.9 Test 8 Double Precision Divide

This test contains 19 subtests which verify the correct execution of the Divide Floating Point Doubleword (DVFD) instruction.

5.10 Test 9 Single Precision Chain

This test verifies the correct functioning of a series of all four single precision instructions.

5.11 Test 10 Double Precision Chain

This test verifies the correct functioning of a series of all four double precision instructions in the same sequence as in Test 9.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould CONCEPT 32/27 Floating Point Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname,MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould CONCEPT 32/27 Floating Point Diagnostic parameters and a summary of their functions follow:

Parameter	Type	Description
A	Available	Available for general use.
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PD directive.
E	Available	Available for general use.
F	Available	Available for general use.
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Available	Available for general use.
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for the JZ directive.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\begin{Bmatrix} n \\ Z \end{Bmatrix}$ /k

J $\begin{Bmatrix} n \\ Z \end{Bmatrix}$ /p

J $\begin{Bmatrix} n \\ Z \end{Bmatrix}$ /p(op)k

J $\begin{Bmatrix} n \\ Z \end{Bmatrix}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L='' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD(op) {k | Pp}

PI(op) {k | Pp}

PZ= {k | Pp}

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould CONCEPT 32/27 Floating Point Diagnostic Unique Directives

The Gould CONCEPT 32/27 Floating Point Diagnostic contains unique directives which perform functions specific to the program.

The Gould CONCEPT 32/27 Floating Point Diagnostic unique directives follow.

6.6.1 "A" Directives

ADFD

Performs a double precision floating point add using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

ADFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

ADFW

Performs a single precision floating point add using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

ADFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.2 "C" Directives

CHFD

Performs a double precision floating point chain using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

CHFD='DST (V_1), SCR (V_2)'

Performs a double precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

CHFW

Performs a single precision floating point chain using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

CHFW='DST(V_1),SCR(V_2)'

Performs a single precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.3 "D" Directives

DVFD

Performs a double precision floating point divide using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

DVFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

DVFW

Performs a single precision floating point divide using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

DVFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.4 "M" Directives

MPFD

Performs a double precision floating point multiply using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

MPFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

MPFW

Performs a single precision floating point multiply using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

MPFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.5 "S" Directives

SUFD

Performs a double precision floating point subtract using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

SUFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

SUFW

Performs a single precision floating point subtract using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

SUFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 must be normalized floating point numbers. Both arguments are optional and may be specified in either order.

Gould 67 Floating Point

(67.FPT.r)

Diagnostic Description

Diagnostic Number: 671032-270r

Documentation Release Level: 1.2

Documentation Release Date: May 1985

Publication Order Number: 326-005120-200

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

CONCEPT/32 is a trademark of Gould Inc.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould 67 Floating Point (67.FPT.r) Diagnostic Description Release 1.0, Publication Order Number **326-005120-000**, was printed September 1984.

The Gould 67 Floating Point (67.FPT.r) Diagnostic Description Release 1.1, Publication Order Number **326-005120-100**, was printed November 1984.

The Gould 67 Floating Point (67.FPT.r) Diagnostic Description Release 1.2, Publication Order Number **326-005120-200**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-2
3-1/3-2
4-1 through 4-2
5-1 through 5-3/5-4
6-1 through 6-10

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Option Set 2-1

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Terminating Program Execution 3-1

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Single Precision Add 5-2
5.3 Test 2 Single Precision Subtract 5-2
5.4 Test 3 Single Precision Multiply 5-2
5.5 Test 4 Single Precision Divide 5-2
5.6 Test 5 Double Precision Add 5-2
5.7 Test 6 Double Precision Subtract 5-2
5.8 Test 7 Double Precision Multiply 5-2
5.9 Test 8 Double Precision Divide 5-2
5.10 Test 9 Single Precision Chained Operations 5-3
5.11 Test 10 Double Precision Chained Operations 5-3
5.12 Test 11 Float Integer Word 5-3
5.13 Test 12 Fix Single Precision Floating Point 5-3
5.14 Test 13 Float Integer Doubleword 5-3
5.15 Test 14 Fix Double Precision Floating Point 5-3

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives. 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould 67 Floating Point Diagnostic Unique Directives 6-7
6.6.1 "A" Directives 6-7
6.6.2 "C" Directives 6-8
6.6.3 "D" Directives 6-8
6.6.4 "F" Directives 6-9
6.6.5 "M" Directives 6-10
6.6.6 "S" Directives 6-10

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould 67 Floating Point Diagnostic (67.FPT.r) is a level two macrodiagnostic program that loads and executes under control of the Gould Diagnostic Executive Program (DEXP).

The Gould 67 Floating Point Diagnostic (67.FPT.r) is valid for the CONCEPT 32/67 and Virtual Memory (V6) processors.

This macrodiagnostic program contains a sequence of 14 tests which verify the correct operation of a floating point instruction or combination of floating point instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives provided by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode).

1.2 Resource Requirements

The Gould 67 Floating Point Diagnostic requires:

- A Gould CPU (either a CONCEPT 32/67 or Virtual Memory-V6 with a minimum of 64K words of memory
- A Floating Point Accelerator (optional)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould 67 Floating Point Diagnostic is a self-initializing program and sets up its default assignment values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the Diagnostic Facility User's Manual. The program name for loading the Gould 67 Floating Point Diagnostic is 67.FPT.r.

2.2 Program Option Set

Once loaded, the Gould 67 Floating Point Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould 67 Floating Point Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.

Option	Default	Effect If Set
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Reserved, do not use.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file after every 5000 passes.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone messages to the LO file.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Unimplemented.
16	Reset	Simulates error conditions. Setting this option simulates all possible error conditions in program tests, allowing a step-through execution. Use this function as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.
17	Reset	Unimplemented.
18	Reset	Inhibits testing during pass 1 of the program.
19	Reset	Inhibits testing during pass 2 of the program.
20	Reset	Enables random operand ranging.
21	Reset	Enables direct operand addressing mode only.
22	Reset	Inhibits testing of Register-Register Floating Point Add, Subtract, Multiply, and Divide instructions.
23	Reset	Enables selecting the simulator used to calculate floating point divide results (when used with Option 24).
24	Reset	Selects the simulator to use for calculating the expected floating point division results as follows: Set selects the hardware floating point accelerator simulator. Reset selects the CPU firmware accelerator. If Option 23 is reset, the program bypasses this option.
25	Reset	Inhibits tests which run with Arithmetic Exception traps enabled.
26	Reset	Inhibits tests which run with Arithmetic Exception traps disabled.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#671032-270r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
FLOATING POINT ACCELERATOR PRESENT AND ENABLED
```

If the program determines that the system has a functioning IPU and neither CPU nor IPU is specified in the DEXP Start command, the program writes the following message to the OC file:

```
CAUTION: WITH TASK RUNNING BIASED "ANY", IT IS NOT POSSIBLE
TO ASCERTAIN WHICH COMPONENT OF THE SYSTEM IS FAILING
IN THE EVENT OF AN ERROR.  SUGGEST RUNNING BIASED
"CPU" OR "IPU" ONLY IF FAULT ISOLATION REQUIRED; [C,H,T]?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould 67 Floating Point Diagnostic (67.FPT.r) writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

```
<<<<'taskname'- - - -hh:mm:ss>>>>
```

taskname	specifies the one to eight character program name as indicated in the load map.
hh:mm:ss	specifies the system elapsed execution time at the point of error.

2. The error message header

```
TEST:ttt,  PASS:n,  PC:cccc
```

ttt	specifies the test and subtest where the error occurred.
n	specifies the current pass count.
ccccc	specifies the hexadecimal memory address of the start of the substest.

3. The erring directive message (for directive tests only)

```
DRTV: directive
```

directive	specifies the erring directive.

4.2 Unique Error Messages

If an error occurs during execution, the program writes an error message with one or more of the following components:

The erring floating point instruction, the destination register used, and the location of the source operand:

```
iiiii  DST:dd,  SRC:sssss
```

iii	specifies the floating point instruction in error.
dd	specifies the destination register of the instruction.
ssss	specifies the source operand location of the instruction. This is either a hexadecimal memory operand address (in the case of a memory reference instruction), or a general purpose register (in the case of a register to register instruction).

The destination and source operands:

```
DST OP:nn...nn
SRC OP:nn...nn
```

DST OP	indicates the destination operand value.
SRC OP	indicates the source operand value.
nn...nn	specifies the hexadecimal value (8 or 16 digits) of the indicated operand.

The expected and actual arithmetic results:

```
EXP CC'S:bbbb ANS:nn..nn
ACT CC'S:bbbb ANS:nn..nn
```

EXP CC'S	indicates the expected condition code value.
ACT CC'S	indicates the actual condition code value.
ANS:	indicates the expected or actual answer.
bbbb	specifies, in binary, the condition code contents.
nn...nn	specifies the hexadecimal value of the indicated result (8 or 16 digits).

If an arithmetic exception error is detected, the program writes the following message:

```
EXP AE TRAP PSD:pppppppp pppppppp
ACT AE TRAP PSW:pppppppp pppppppp
```

EXP AE TRAP PSD	indicates the expected arithmetic exception trap program status doubleword.
ACT AE TRAP PSD	indicates the actual arithmetic exception trap program status doubleword.
pppppppp pppppppp	specifies the hexadecimal contents of the trap program status doubleword, or if all zeros, indicates no trap.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The 67 Floating Point Diagnostic (67.FPT.r) contains 14 tests which verify the correct operation of one of the floating point instructions or, in the case of Tests 9 and 10, a chained sequence of back-to-back floating point instructions.

Pass #	Tests	Operand Type	Inhibiting Option (if set)
1	1-14	tests boundary conditions	18
2	1-10	general purpose	19
	11-14	no operations performed	
3	1-14	pseudo-random	—

The program calculates the expected result of each floating point instruction. During floating point division instruction tests, the program chooses between two software simulators: one to predict the results calculated by the CPU and the other to predict the results calculated by the hardware Floating Point Accelerator. The program selects the appropriate simulator on the basis of the status returned in Bit 22 by a Read CPU status (RDSTS) instruction. Set Option 23 (enables selector override) and Option 24 (selects the simulator) to override the selection. The program reports the algorithm in use to the listed output (LO) file after making the first selection, and at any time the selection changes.

The program uses software floating point simulators to calculate the software expected results. Following the execution of each floating point instruction, the program compares the actual hardware result (data, condition codes, and exceptional condition traps) to the software calculated result, and if different, it writes an error message to the LO file.

5.2 Test 1 Single Precision Add

This test contains 17 subtests which verify the correct execution of the Add Floating Point Word (ADFW) and Add Register Floating Point Word (ADRFW) instructions.

5.3 Test 2 Single Precision Subtract

This test contains 17 subtests which verify the correct execution of the Subtract Floating Point Word (SUFW) and Subtract Register Floating Point Word (SURFW) instructions.

5.4 Test 3 Single Precision Multiply

This test contains 17 subtests which verify the correct execution of the Multiply Floating Point Word (MPFW) and Multiply Register Floating Point Word (MPRFW) instructions.

5.5 Test 4 Single Precision Divide

This test contains 17 subtests which verify the correct execution of the Divide Floating Point Word (DVFW) and Divide Register Floating Point Word (DVRFW) instructions.

5.6 Test 5 Double Precision Add

This test contains 26 subtests which verify the correct execution of the Add Floating Point Doubleword (ADFD) and Add Register Floating Point Doubleword (ADRFD) instructions.

5.7 Test 6 Double Precision Subtract

This test contains 26 subtests which verify the correct execution of the Subtract Floating Point Doubleword (SUFD) and Subtract Register Floating Point Doubleword (SURFD) instructions.

5.8 Test 7 Double Precision Multiply

This test contains 26 subtests which verify the correct execution of the Multiply Floating Point Doubleword (MPFD) and Multiply Register Floating Point Doubleword (MPRFD) instructions.

5.9 Test 8 Double Precision Divide

This test contains 26 subtests which verify the correct execution of the Divide Floating Point Doubleword (DVFD) and Divide Register Floating Point Doubleword (DVRFD) instructions.

5.10 Test 9 Single Precision Chained Operations

This test contains two subtests which verify the correct functioning of a series of all four single precision instructions.

5.11 Test 10 Double Precision Chained Operations

This test contains two subtests which verify the correct functioning of a series of all four double precision instructions in the same sequence as in Test 9.

5.12 Test 11 Float Integer Word

This test contains six subtests which verify the correct execution of the Float Integer Word (FLTW) instruction.

5.13 Test 12 Fix Single Precision Floating Point

This test contains six subtests which verify the correct execution of the Fix Floating Point Word (FIXW) instruction.

5.14 Test 13 Float Integer Doubleword

This test contains six subtests which verify the correct execution of the Float Integer Doubleword (FLTD) instruction.

5.15 Test 14 Fix Double Precision Floating Point

This test contains six subtests which verify the correct execution of the Fix Floating Point Doubleword (FIXD) instruction.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould 67 Floating Point Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould 97 Floating Point Diagnostic parameters and a summary of their functions follow:

Parameter	Use	Description
A	Available	Available for general use.
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PD directive.
E	Available	Available for general use.
F	Available	Available for general use.
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Available	Available for general use.
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for the JZ directive.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L=' ' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD (op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PI(op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PZ= $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z)
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn — Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME
TIME(message) — Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould 67 Floating Point Diagnostic Unique Directives

The Gould 67 Floating Point Diagnostic contains unique directives which perform functions specific to the program.

The Gould 67 Floating Point Diagnostic unique directives follow.

6.6.1 "A" Directives

ADFD

Performs a double precision floating point add using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

ADFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

ADFW

Performs a single precision floating point add using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

ADFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.2 "C" Directives

CHFD

Performs a double precision floating point chain using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

CHFD='DST (V_1), SCR (V_2)'

Performs a double precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

CHFW

Performs a single precision floating point chain using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

CHFW='DST(V_1),SCR(V_2)'

Performs a single precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.3 "D" Directives

DVFD

Performs a double precision floating point divide during the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

DVFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

DVFW

Performs a single precision floating point divide using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

DVFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2, as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.4 "F" Directives

FIXD

Performs a double precision conversion from floating point to integer using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FIXD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FIXW

Performs a single precision conversion from floating point to integer using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FIXW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FLTD

Performs a double precision conversion from integer to floating point using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FLTD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

FLTW

Performs a single precision conversion from integer to floating point using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FLTW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

6.6.5 "M" Directives

MPFD

Performs a double precision floating point multiply using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

MPFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

MPFW

Performs a single precision floating point multiply using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

MPFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values V_1 and V_2 are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.6 "S" Directives

SUFD

Performs a double precision floating point subtract using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

SUFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

SUFW

Performs a single precision floating point subtract using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

SUFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

Gould CONCEPT 32/87

Floating Point

(87_FPT.r)

Diagnostic Description

Diagnostic Number: 871032-270r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005710-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

CONCEPT/32 is a trademark of Gould Inc.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould CONCEPT 32/87 Floating Point (87.FPT.r) Diagnostic Description Release 1.0, Publication Order Number **326-005710-000**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-2
3-1/3-2
4-1 through 4-2
5-1 through 5-3/5-4
6-1 through 6-10

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Option Set 2-1

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Terminating Program Execution 3-1

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Single Precision Add 5-2
5.3 Test 2 Single Precision Subtract 5-2
5.4 Test 3 Single Precision Multiply 5-2
5.5 Test 4 Single Precision Divide 5-2
5.6 Test 5 Double Precision Add 5-2
5.7 Test 6 Double Precision Subtract 5-2
5.8 Test 7 Double Precision Multiply 5-2
5.9 Test 8 Double Precision Divide 5-2
5.10 Test 9 Single Precision Chain 5-2
5.11 Test 10 Double Precision Chain 5-3
5.12 Test 11 Single Precision Float 5-3
5.13 Test 12 Single Precision Fix 5-3
5.14 Test 13 Double Precision Float 5-3
5.15 Test 14 Double Precision Fix 5-3

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould CONCEPT 32/87 Floating Point Diagnostic Unique Directives 6-7
6.6.1 "A" Directives 6-7
6.6.2 "C" Directives 6-8
6.6.3 "D" Directives 6-8
6.6.4 "F" Directives 6-9
6.6.5 "M" Directives 6-10
6.6.6 "S" Directives 6-10

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould CONCEPT 32/87 Floating Point Diagnostic (87.FPT.r) is a level two macrodiagnostic program that loads and executes under control of the CONCEPT/32 Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 14 tests which verify the correct operation of a floating point instruction or combination of floating point instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode).

1.2 Resource Requirements

The Gould CONCEPT 32/87 Floating Point Diagnostic requires:

- A CONCEPT 32/87 CPU with a minimum of 64K words of memory
- A Multiply Accelerator (optional)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould CONCEPT 32/87 Floating Point Diagnostic is a self-initializing program and sets up its default assignment values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the Diagnostic Facility User's Manual. The program name for loading the Gould CONCEPT 32/87 Floating Point Diagnostic is 87.FPT.r.

2.2 Program Option Set

Once loaded, the Gould CONCEPT 32/87 Floating Point Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould CONCEPT 32/87 Floating Point Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T] ? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.

Option	Default	Effect If Set
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Reserved, do not use.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file after every 5000 passes.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Unimplemented.
16	Reset	Simulates error conditions. Setting this option simulates all possible error conditions in program tests, allowing a step-through execution. Use this function as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.
17	Reset	Unimplemented.
18	Reset	Inhibits testing during pass 1 of the program.
19	Reset	Inhibits testing during pass 2 of the program.
20	Reset	Enables random operand ranging.
21	Reset	Enables direct operand addressing mode only.
22	Reset	Inhibits testing of Register-Register Floating Point Add, Subtract, Multiply, and Divide instructions.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#871032-270r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

If the program determines that the system has a functioning IPU and neither CPU nor IPU is specified in the DEXP Start command, the program writes the following message to the OC file:

```
CAUTION:  WITH TASK RUNNING BIASED "ANY", IT IS NOT POSSIBLE TO
ASCERTAIN WHICH COMPONENT OF THE SYSTEM IS FAILING IN THE EVENT
OF AN ERROR.  SUGGEST RUNNING BIASED "CPU" OR "IPU" ONLY IF FAULT
ISOLATION REQUIRED; [C,H,T] ?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP>prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould CONCEPT 32/87 Floating Point Diagnostic (87.FPT.r) writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<<  'taskname'----hh:mm:ss  >>>>
   ```

   taskname — specifies the one to eight character program name as indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TEST:ttt,  PASS:n,  PC:ccccc
   ```

   ttt — specifies the test and subtest where the error occurred.

   n — specifies the current pass count.

   ccccc — specifies the hexadecimal memory address of the start of the subtest.

3. The erring directive message (for directive tests only)

   ```
   DRTV:directive
   ```

   directive — specifies the erring directive.

4.2 Unique Error Messages

If an error occurs during execution, the program writes an error message with one or more of the following components:

The erring floating point instruction, the destination register used, and the location of the source operand:

```
iiiii  DST:dd,  SRC:sssss
```

iiiii — specifies the floating point instruction in error.

dd — specifies the destination register of the instruction.

sssss — specifies the location of the source operand of the instruction. This is either a hexadecimal memory operand address (in the case of a memory reference instruction), or a general purpose register (in the case of a register to register instruction).

The destination and source operands:

```
DST OP:nn...nn
SRC OP:nn...nn
```

DST OP — indicates the destination operand value.

SRC OP — indicates the source operand value.

nn...nn — specifies the hexadecimal value (8 or 16 digits) of the indicated operand.

The expected and actual arithmetic results:

```
EXP CC'S:bbbb ANS:nn..nn
ACT CC'S:bbbb ANS:nn..nn
```

EXP CC'S — indicates the expected condition code value.

ACT CC'S — indicates the actual condition code value.

ANS: — indicates the expected or actual answer.

bbbb — specifies, in binary, the condition code contents.

nn...nn — specifies the hexadecimal value of the indicated result (8 or 16 digits).

If an arithmetic exception error is detected, the program writes the following message:

```
EXP AE TRAP PSD:pppppppp pppppppp
ACT AE TRAP PSW:pppppppp pppppppp
```

EXP AE TRAP PSD — indicates the expected arithmetic exception trap program status doubleword.

ACT AE TRAP PSD — indicates the actual arithmetic exception trap program status doubleword.

pppppppp pppppppp — specifies the hexadecimal contents of the trap program status doubleword, or if all zeros, indicates no trap.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould CONCEPT 32/87 Floating Point Diagnostic (87.FPT.r) contains 14 test sequences which verify the correct operation of one of the floating point instuctions or, in the case of Tests 9 and 10, a chained sequence of back-to-back floating point instructions.

Pass #	Tests	Operand Type	Inhibiting Option (if set)
1	1-14	tests boundary conditions	18
2	1-10	general purpose	19
	11-14	no operations performed	
3	1-14	pseudo - random	-

The program uses software floating point simulators to calculate the software expected results. After the execution of each floating point instruction, the program compares the actual hardware result (data, condition codes, and exceptional condition traps) to the software calculated result, and if different, it writes an error message to the listed output (LO) file.

Note: If, during execution of pass 2, the program determines that a Multiply Accelerator (MACC) is present and functioning, it writes the following message to the LO file after Tests 3 and 7 and executes boundary condition MACC operands:

hh:mm:ss 'taskname' (MULTIPLY ACCELERATOR PRESENT)

If a Multiply Accelerator (MACC) is not present, the program fails to write the message to the LO file but executes boundary condition MACC operands.

5.2 Test 1 Single Precision Add

This test contains 16 subtests which verify the correct execution of the Add Floating Point Word (ADFW) and Add Register Floating Point Word (ADRFW) instructions.

5.3 Test 2 Single Precision Subtract

This test contains 16 subtests which verify the correct execution of the Subtract Floating Point Word (SUFW) and Subtract Register Floating Point Word (SURFW) instructions.

5.4 Test 3 Single Precision Multiply

This test contains 16 subtests which verify the correct execution of the Multiply Floating Point Word (MPFW) and Multiply Register Floating Point Word (MPRFW) instructions.

5.5 Test 4 Single Precision Divide

This test contains 16 subtests which verify the correct execution of the Divide Floating Point Word (DVFW) and Divide Register Floating Point Word (DVRFW) instructions.

5.6 Test 5 Double Precision Add

This test contains 25 subtests which verify the correct execution of the Add Floating Point Doubleword (ADFD) and Add Register Floating Point Doubleword (ADRFD) instructions.

5.7 Test 6 Double Precision Subtract

This test contains 25 subtests which verify the correct execution of the Subtract Floating Point Doubleword (SUFD) and Subtract Register Floating Point Doubleword (SURFD) instructions.

5.8 Test 7 Double Precision Multiply

This test contains 25 subtests which verify the correct execution of the Multiply Floating Point Doubleword (MPFD) and Multiply Register Floating Point Doubleword (MPRFD) instructions.

5.9 Test 8 Double Precision Divide

This test contains 25 subtests which verify the correct execution of the Divide Floating Point Doubleword (DVFD) and Divide Register Floating Point Doubleword (DVRFD) instructons.

5.10 Test 9 Single Precision Chain

This test contains two subtests which verify the correct functioning of a series of all four single precision instructions.

5.11 Test 10 Double Precision Chain

This test contains two subtests which verify the correct functioning of a series of all four double precision instructions in the same sequence as in Test 9.

5.12 Test 11 Single Precision Float

This test contains six subtests which verify the correct execution of the Float Integer Word (FLTW) instruction.

5.13 Test 12 Single Precision Fix

This test contains six subtests which verify the correct execution of the Fix Floating Point Word (FIXW) instruction.

5.14 Test 13 Double Precision Float

This test contains six subtests which verify the correct execution of the Float Integer Doubleword (FLTD) instruction.

5.15 Test 14 Double Precision Fix

This test contains six subtests which verify the correct execution of the Fix Floating Point Doubleword (FIXD) instruction.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould CONCEPT 32/87 Floating Point Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

```
'taskname ACTIVE [C,T] ?
```

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

```
DXP>taskname LA,CR,DO <cr>
```

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould CONCEPT 32/87 Floating Point Diagnostic parameters and a summary of their functions follow:

Parameter	Type	Description
A	Available	Available for general use.
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PD directive.
E	Available	Available for general use.
F	Available	Available for general use.
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Available	Available for general use.
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for the JZ directive.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L=' 'or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

P_p (op) k

Pp_1 (op) Pp_2

PD (op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PI (op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PZ = $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould CONCEPT 32/87 Floating Point Diagnostic Unique Directives

The Gould CONCEPT 32/87 Floating Point Diagnostic contains unique directives which perform functions specific to the program.

The Gould CONCEPT 32/87 Floating Point Diagnostic unique directives follow.

6.6.1 "A" Directives

ADFD

Performs a double precision floating point add using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

ADFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

ADFW

Performs a single precision floating point add using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

ADFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.2 "C" Directives

CHFD

Performs a double precision floating point chain using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

CHFD='DST(V_1),SCR(V_2)'

Performs a double precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

CHFW

Performs a single precision floating point chain using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

CHFW='DST(V_1),SCR(V_2)'

Performs a single precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.3 "D" Directives

DVFD

Performs a double precision floating point divide using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

DVFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

DVFW

Performs a single precision floating point divide using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

DVFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.4 "F" Directives

FIXD

Performs a double precision conversion from floating point to integer using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FIXD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FIXW

Performs a single precision conversion from floating point to integer using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FIXW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FLTD

Performs a double precision conversion from integer to floating point using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FLTD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

FLTW

Performs a single precision conversion from integer to floating point using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FLTW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

6.6.5 "M" Directives

MPFD

Performs a double precision floating point multiply using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

MPFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

MPFW

Performs a single precision floating point multiply using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

MPFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.6 "S" Directives

SUFD

Performs a double precision floating point subtract using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

SUFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

SUFW

Performs a single precision floating point subtract using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

SUFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

Gould 97 Floating Point

(97.FPT.r)

Diagnostic Description

Diagnostic Number: 971032-270r

Documentation Release Level: 1.1

Documentation Release Date: May 1985

Publication Order Number: 326-005470-100

GOULD
Electronics

This manual is supplied without representation of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

CONCEPT/32 is a trademark of Gould Inc.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould 97 Floating Point (97.FPT.r) Diagnostic Description Release 1.0, Publication Order Number **326-005470-000**, was printed November 1984.

The Gould 97 Floating Point (97.FPT.r) Diagnostic Description Release 1.1, Publication Order Number **326-005470-100**, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vi
1-1/1-2
2-1 through 2-2
3-1/3-2
4-1 through 4-2
5-1 through 5-3/5-4
6-1 through 6-10

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Option Set 2-1

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Terminating Program Execution 3-1

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Single Precision Add 5-2
5.3 Test 2 Single Precision Subtract 5-2
5.4 Test 3 Single Precision Multiply 5-2
5.5 Test 4 Single Precision Divide 5-2
5.6 Test 5 Double Precision Add 5-2
5.7 Test 6 Double Precision Subtract 5-2
5.8 Test 7 Double Precision Multiply 5-2
5.9 Test 8 Double Precision Divide 5-2
5.10 Test 9 Single Precision Chain 5-2
5.11 Test 10 Double Precision Chain 5-3
5.12 Test 11 Single Precision Float 5-3
5.13 Test 12 Single Precision Fix 5-3
5.14 Test 13 Double Precision Float 5-3
5.15 Test 14 Double Precision Fix 5-3

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould 97 Floating Point Diagnostic Unique Directives 6-7
6.6.1 "A" Directives 6-7
6.6.2 "C" Directives 6-8
6.6.3 "D" Directives 6-8
6.6.4 "F" Directives 6-9
6.6.5 "M" Directives 6-10
6.6.6 "S" Directives 6-10

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould 97 Floating Point Diagnostic (97.FPT.r) is a level two macrodiagnostic program that loads and executes under control of the Gould Diagnostic Executive Program (DEXP).

The Gould 97 Floating Point Diagnostic (97.FPT.r) is valid for the CONCEPT 32/97 and Virtual Memory (V9) processors.

This macrodiagnostic program contains a sequence of 14 tests which verify the correct operation of a floating point instruction or combination of floating point instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution the program writes pass count messages, milestone messages, and error messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the OC file (if in Interactive mode).

1.2 Resource Requirements

The Gould 97 Floating Point Diagnostic requires:

- A Gould CPU (either a CONCEPT 32/97 or a Virtual Memory-V9 with a minimum of 64K words of memory
- A Multiply Accelerator (optional)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould 97 Floating Point Diagnostic is a self-initializing program and sets up its default assignment values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the Diagnostic Facility User's Manual. The program name for loading the Gould 97 Floating Point Diagnostic is 97.FPT.r.

2.2 Program Option Set

Once loaded, the Gould 97 Floating Point Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould 97 Floating Point Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends program execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.

Option	Default	Effect If Set
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Reserved, do not use.
8	Reset	Unimplemented.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes program's pass count messages to the listed output (LO) file after every 5000 passes.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Unimplemented.
16	Reset	Simulates error conditions. Setting this option simulates all possible error conditions in program tests, allowing a step-through execution. Use this function as a maintenance or debugging aid by verifying and demonstrating test operations and error message routines.
17	Reset	Unimplemented.
18	Reset	Inhibits testing during pass 1 of the program.
19	Reset	Inhibits testing during pass 2 of the program.
20	Reset	Enables random operand ranging.
21	Reset	Enables direct operand addressing mode only.
22	Reset	Inhibits testing of Register-Register Floating Point Add, Subtract, Multiply, and Divide instructions.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#971032-270r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
FLOATING POINT ACCELERATOR PRESENT AND ENABLED
```

If the program determines the system has a functioning IPU and neither CPU nor IPU is specified in the DEXP Start command, the program writes the following message to the OC file:

```
CAUTION: WITH TASK RUNNING BIASED "ANY", IT IS NOT POSSIBLE
TO ASCERTAIN WHICH COMPONENT OF THE SYSTEM IS FAILING
IN THE EVENT OF AN ERROR. SUGGEST RUNNING BIASED
"CPU" OR "IPU" ONLY IF FAULT ISOLATION REQUIRED; [C,H,T] ?
```

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould 97 Floating Point Diagnostic (97.FPT.r) writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<< 'taskname'----hh:mm:ss >>>>
   ```

   taskname — specifies the one to eight character program name as indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TEST:ttt,  PASS:n,  PC:ccccc
   ```

   ttt — specifies the test and subtest where the error occurred.

   n — specifies the current pass count.

   ccccc — specifies the hexadecimal memory address of the start of the subtest.

3. The erring directive message (for directive tests only)

   ```
   DRTV:directive
   ```

   directive — specifies the erring directive.

4.2 Unique Error Messages

If an error occurs during execution, the program writes an error message with one or more of the following components:

The erring floating point instruction, the destination register used, and the location of the source operand

```
iiiii  DST:dd,  SRC:sssss
```

iiiii — specifies the floating point instruction in error.

dd — specifies the destination register of the instruction.

sssss — specifies the location of the source operand of the instruction. This is either a hexadecimal memory operand address (in the case of a memory reference instruction), or a general purpose register (in the case of a register to register instruction).

The destination and source operands

```
DST OP:nn...nn
SRC OP:nn...nn
```

DST OP — indicates the destination operand value.

SRC OP — indicates the source operand value.

nn...nn — specifies the hexadecimal value (8 or 16 digits) of the operand indicated.

The expected and actual arithmetic results

```
EXP CC'S:bbbb ANS:nn..nn
ACT CC'S:bbbb ANS:nn..nn
```

EXP CC'S — indicates the expected condition code value.

ACT CC'S — indicates the actual condition code value.

ANS: — indicates the expected or actual answer.

bbbb — specifies, in binary, the condition code contents.

nn...nn — specifies the hexadecimal value of the indicated result (8 or 16 digits).

If an arithmetic exception error is detected, the program writes the following message:

```
EXP AE TRAP PSD:pppppppp pppppppp
ACT AE TRAP PSW:pppppppp pppppppp
```

EXP AE TRAP PSD — indicates the expected arithmetic exception trap program status doubleword.

ACT AE TRAP PSD — indicates the actual arithmetic exception trap program status doubleword.

pppppppp pppppppp — specifies the hexadecimal contents of the trap program status doubleword, or if all zeros, indicates no trap.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould 97 Floating Point Diagnostic ('97.FPT.r) contains 14 tests which verify the correct operation of one of the floating point instructions or, in the case of Tests 9 and 10, a chained sequence of back-to-back floating point instructions.

Pass #	Tests	Operand Type	Inhibiting Option (if set)
1	1-14	tests boundary conditions	18
2	1-10	general purpose	19
	11-14	no operations performed	
3	1-14	pseudo - random	-

The program calculates the expected result of each floating point instruction. During floating point division instruction tests, the program chooses between two software simulators: one to predict the results calculated by the CPU and the other to predict the results calculated by the hardware Floating Point Accelerator. The program selects the appropriate simulator on the basis of the status returned in Bit 22 by a Read CPU status (RDSTS) instruction. The program reports the algorithm in use to the listed output (LO) file after making the first selection and any time the selection changes.

The program uses software floating point simulators to calculate the software expected results. Following the execution of each floating point instruction, the program compares the actual hardware result (data, condition codes, and exceptional condition traps) to the software calculated result, and if different, it writes an error message to the LO file.

Note: If, during execution of pass 2, the program determines a Multiply Accelerator (MACC) is present and functioning, it writes the following message to the LO file after Tests 3 and 7, and executes boundary condition MACC operands:

hh:mm:ss 'taskname' (MULTIPLY ACCELERATOR PRESENT)

If a Multiply Accelerator (MACC) is not present, the program fails to write a message but executes boundary condition MACC operands.

5.2 Test 1 Single Precision Add

This test contains 16 subtests which verify the correct execution of the Add Floating Point Word (ADFW) and Add Register Floating Point Word (ADRFW) instructions.

5.3 Test 2 Single Precision Subtract

This test contains 16 subtests which verify the correct execution of the Subtract Floating Point Word (SUFW) and Subtract Register Floating Point Word (SURFW) instructions.

5.4 Test 3 Single Precision Multiply

This test contains 16 subtests which verify the correct execution of the Multiply Floating Point Word (MPFW) and Multiply Register Floating Point Word (MPRFW) instructions.

5.5 Test 4 Single Precision Divide

This test contains 16 subtests which verify the correct execution of the Divide Floating Point Word (DVFW) and Divide Register Floating Point Word (DVRFW) instructions.

5.6 Test 5 Double Precision Add

This test contains 25 subtests which verify the correct execution of the Add Floating Point Doubleword (ADFD) and Add Register Floating Point Doubleword (ADRFD) instructions.

5.7 Test 6 Double Precision Subtract

This test contains 25 subtests which verify the correct execution of the Subtract Floating Point Doubleword (SUFD) and Subtract Register Floating Point Doubleword (SURFD) instructions.

5.8 Test 7 Double Precision Multiply

This test contains 25 subtests which verify the correct execution of the Multiply Floating Point Doubleword (MPFD) and Multiply Register Floating Point Doubleword (MPRFD) instructions.

5.9 Test 8 Double Precision Divide

This test contains 25 subtests which verify the correct execution of the Divide Floating Point Doubleword (DVFD) and Divide Register Floating Point Doubleword (DVRFD) instructions.

5.10 Test 9 Single Precision Chain

This test contains two subtests which verify the correct functioning of a series of all four single precision instructions.

5.11 Test 10 Double Precision Chain

This test contains two subtests which verify the correct functioning of a series of all four double precision instructions in the same sequence as in Test 9.

5.12 Test 11 Single Precision Float

This test contains six subtests which verify the correct execution of the Float Integer Word (FLTW) instruction.

5.13 Test 12 Single Precision Fix

This test contains six subtests which verify the correct execution of the Fix Floating Point Word (FIXW) instruction.

5.14 Test 13 Double Precision Float

This test contains six subtests which verify the correct execution of the Float Integer Doubleword (FLTD) instruction.

5.15 Test 14 Double Precision Fix

This test contains six subtests which verify the correct execution of the Fix Floating Point Doubleword (FIXD) instruction.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould 97 Floating Point Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould 97 Floating Point Diagnostic parameters and a summary of their functions follow:

Parameter	Use	Description
A	Available	Available for general use.
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PD directive.
E	Available	Available for general use.
F	Available	Available for general use.
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Available	Available for general use.
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Available	Available for general use.
T	Available	Available for general use.
U	Available	Available for general use.
V	Available	Available for general use.
W	Available	Available for general use.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for the JZ directive.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L=' ' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD (op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

PI(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

PZ= $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$

p	specifies one of the 26 parameters (identified A through Z)
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn — Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME
TIME(message) — Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould 97 Floating Point Diagnostic Unique Directives

The Gould 97 Floating Point Diagnostic contains unique directives which perform functions specific to the program.

The Gould 97 Floating Point Diagnostic unique directives follow.

6.6.1 "A" Directives

ADFD

Performs a double precision floating point add using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

ADFD='DST(V_1,SRC(V_2)'

Performs a double precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

ADFW

Performs a single precision floating point add using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

ADFW='DST(V_1,SRC(V_2)'

Performs a single precision floating point add using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.2 "C" Directives

CHFD

Performs a double precision floating point chain using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

CHFD='DST (V_1), SCR (V_2)'

Performs a double precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

CHFW

Performs a single precision floating point chain using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

CHFW='DST(V_1),SCR(V_2)'

Performs a single precision floating point chain using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. If either of the values, V_1 or V_2, is an unnormalized floating point number, the other value must be zero. Both arguments are optional and may be specified in either order.

6.6.3 "D" Directives

DVFD

Performs a double precision floating point divide using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

DVFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

DVFW

Performs a single precision floating point divide using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

DVFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point divide using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.4 "F" Directives

FIXD

Performs a double precision conversion from floating point to integer using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FIXD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FIXW

Performs a single precision conversion from floating point to integer using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FIXW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from floating point to integer using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. The value V_2 is assumed to be a normalized floating point number. Both arguments are optional and may be specified in either order.

FLTD

Performs a double precision conversion from integer to floating point using the value in doubleword DSTOP as the initial value of the destination register and the value in doubleword SRCOP as the source operand.

FLTD='DST(V_1),SRC(V_2)'

Performs a double precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

FLTW

Performs a single precision conversion from integer to floating point using the value in word DSTOP as the initial value of the destination register and the value in word SRCOP as the source operand.

FLTW='DST(V_1),SRC(V_2)'

Performs a single precision conversion from integer to floating point using the specified value V_1 as the initial value of the destination register and the specified value V_2 as the source operand. Both arguments are optional and may be specified in either order.

6.6.5 "M" Directives

MPFD

Performs a double precision floating point multiply using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

MPFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

MPFW

Performs a single precision floating point multiply using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

MPFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point multiply using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

6.6.6 "S" Directives

SUFD

Performs a double precision floating point subtract using the value in doubleword DSTOP as the destination operand and the value in doubleword SRCOP as the source operand.

SUFD='DST(V_1),SRC(V_2)'

Performs a double precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

SUFW

Performs a single precision floating point subtract using the value in word DSTOP as the destination operand and the value in word SRCOP as the source operand.

SUFW='DST(V_1),SRC(V_2)'

Performs a single precision floating point subtract using the specified value V_1 as the destination operand and the specified value V_2 as the source operand. The values, V_1 and V_2, are assumed to be normalized floating point numbers. Both arguments are optional and may be specified in either order.

Gould Ethernet Controller

(EC_ENT.r)

Diagnostic Description

Diagnostic Number: 001032-988r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005650-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould Ethernet Controller (EC.ENT.r) Diagnostic Description Release 1.0, Publication Order Number 326-005650-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vii/viii
1-1/1-2
2-1 through 2-6
3-1 through 3-2
4-1 through 4-13/4-14
5-1 through 5-10
6-1 through 6-18
7-1 through 7-5/7-6

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Initialization 2-1
2.3 Program Option Set 2-5

CHAPTER 3 PROGRAM EXECUTION

3.1 Starting Program Execution 3-1
3.2 Milestone Messages 3-1
3.3 Terminating Program Execution 3-2

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Message Routines 4-1
4.1.1 The STARTIO Routine and Error Message 4-3
4.1.2 The WAIT4SI Routine and Error Messages 4-4
4.1.3 The COPYSTAT Routine and Error Messages 4-5
4.1.4 The BUFFCOMP Routine and Error Message 4-7
4.1.5 The ENDERMSG Routine and Error Message 4-8
4.2 Unique Messages 4-9
4.3 Permanent Macro Recovery Routines 4-12
4.4 Initialization Error Messages 4-12
4.5 Run-time Error Codes 4-13
4.6 Queue-drive Error Codes 4-13

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test Execution 5-2
5.3 Test 1 No Operation 5-3
5.4 Test 2 Reset Controller and Sense Command 5-3
5.5 Test 3 Invalid Commands 5-3
5.6 Test 4 Program Check Test 5-3

5.7 Test 5 Load Individual Address 5-3
5.8 Test 6 Transfer-in-Channel 5-3
5.9 Test 7 No Operation With Post Program Controlled Interrupt 5-3
5.10 Test 8 Halt I/O (No I/O in Progress) 5-4
5.11 Test 9 Read Statistics Data Chaining 5-4
5.12 Test 10 Stop I/O 5-4
5.13 Test 11 Incorrect Length 5-4
5.14 Test 12 Subchannel Busy 5-4
5.15 Test 13 Invalid Instruction 5-4
5.16 Test 14 Sense Command With Skip Flag 5-4
5.17 Test 15 Halt I/O (Read in Progress) 5-4
5.18 Test 16 Halt I/O (Write in Progress) 5-4
5.19 Test 17 Promiscuous Mode (Loopback) 5-5
5.20 Test 18 Broadcast Address Recognition (Loopback) 5-5
5.21 Test 19 Read Statistics and Sense (Loopback) 5-5
5.22 Test 20 Varying Transfer Count (Loopback) 5-5
5.23 Test 21 Read Data Chaining (Loopback) 5-5
5.24 Test 22 Write Data Chaining (Loopback) 5-5
5.25 Test 23 Read and Write Data Chaining (Loopback) 5-5
5.26 Test 24 Write Data Chained With Byte Addresses 5-5
5.27 Test 25 Read With Invalid Address (Loopback) 5-5
5.28 Test 26 Read and Write Command Chaining (Loopback) 5-6
5.29 Test 27 Transfer-in-Channel (Loopback) 5-6
5.30 Test 28 Post Program Controlled Interrupt (Loopback) 5-6
5.31 Test 29 Read Variable Length Frames From Local Storage (Loopback) 5-6
5.32 Test 30 Read Multiple Frames From Local Storage (Loopback) 5-6
5.33 Test 31 Long Frame Status (Loopback) 5-6
5.34 Test 32 Short Frame Status (Loopback) 5-7
5.35 Test 33 Multicast Address Recognition And Disable Multicast Frames (Loopback) 5-7
5.36 Test 34 Read With Skip Flag (Loopback) 5-7
5.37 Test 35 Transmit and Receive Data On Same Subchannel (Loopback) 5-7
5.38 Test 36 Local Buffer Overflow Error (Loopback) 5-7
5.39 Test 37 Nonpresent Memory Access 5-7
5.40 Test 38 Ping Pong Mode 5-8
5.41 Test 39 Varying Transfer Counts 5-8
5.42 Test 40 Read Data Chaining 5-8
5.43 Test 41 Write Data Chaining 5-8
5.44 Test 42 Read and Write Data Chaining 5-8
5.45 Test 43 Read Command Chaining 5-8
5.46 Test 44 Write Command Chaining 5-8
5.47 Test 45 Read and Write Command Chaining 5-8
5.48 Test 46 Transfer-in-Channel (TIC) 5-8
5.49 Test 47 Post Program Controlled Interrupt Flag Set 5-9
5.50 Test 48 Read Variable Length Frames From Local Storage 5-9
5.51 Test 49 Read Multiple Frames From Local Stoarge 5-9

5.52 Test 50 Long Frame Status 5-9
5.53 Test 51 Short Frame Status 5-9
5.54 Test 52 Multicast Address Recognition And Disable Multicast Address 5-9
5.55 Test 53 Read With Skip Flag 5-9
5.56 Test 54 Local Buffer Overflow 5-9
5.57 Test 55 Contention 5-9
5.58 Test 56 Promiscuous Mode 5-10
5.59 Test 57 Broadcast Address Recognition 5-10
5.60 Test 58 Destination Address Recognition Frame Reject 5-10

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould Ethernet Controller Diagnostic Unique Directives 6-7
6.6.1 "A" Directive 6-7
6.6.2 "B" Directives 6-7
6.6.3 "C" Directives 6-8
6.6.4 "D" Directives 6-9
6.6.5 "E" Directive 6-13
6.6.6 "H" Directives 6-13
6.6.7 "I" Directives 6-13
6.6.8 "L" Directives 6-14
6.6.9 "M" Directives 6-15
6.6.10 "N" Directive 6-15
6.6.11 "R" Directives 6-16
6.6.12 "S" Directives 6-17
6.6.13 "T" Directives 6-18
6.6.14 "W" Directives 6-18
6.6.15 "Z" Directives 6-18

CHAPTER 7 I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview 7-1
7.2 I/O Command Trace 7-1
7.2.1 Starting Execution 7-1
7.2.2 Milestone Messages 7-2
7.2.3 Terminating I/O Command Trace 7-3
7.3 Data Recorder 7-4
7.3.1 Starting Execution 7-4
7.3.2 Milestone Messages 7-4
7.3.3 Terminating Data Recorder 7-5

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould Ethernet Controller (EC.ENT.r) is a level two macrodiagnostic program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 58 tests which verify the correct operation of specific ethernet controller functions. This program tests the ethernet controller in one of the following modes:

- Internal in Local mode (Tests 1 through 37 execute automatically)
- Slave in Remote mode (Tests 38 through 55 execute automatically)
- Master in Remote mode (Tests 38 through 55 execute automatically)

Tests 56 through 58 execute individually (by operator request) and only in the remote modes.

During execution, the program writes error and milestone messages to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode). The CC file assignments are to disc or tape and the OC file assignments are to the operator's console or a terminal configured on an eight-line asynchronous controller.

1.2 Resource Requirements

The Gould Ethernet Controller Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory
- An Ehternet Controller

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

By entering the Return key in response to the prompts, the Gould Ethernet Controller Diagnostic assigns the following default values during initialization:

- Configuration is Internal
- Channel Address is 0C
- Device Subaddresses are 0, 8
- Individual Address is 123456789ABC
- Number of Multicast Addresses to load is 1
- Multicast Address is 010000000123
- Destination Address is 123456789ABC
- Multicast Address for Destination Station is 010000000123

To modify the default values, respond to the initialization prompts with one of the responses written with the prompt or use the DEXP Assign command.

Note: Values modified during initialization or by an Assign command become the default values for all reinitialization sequences until reloading and initializing the program.

2.2 Program Initialization

Before initializing the Gould Ethernet Controller Diagnostic, load it into the system by following the loading procedures in the appropriate Diagnostic Facility User's Manual. The program name for loading the Gould Ethernet Controller Diagnostic is EC.ENT.r. After the program loads, it writes the following initialization sequence to the operator communications (OC) file:

Note: The program reissues any incorrectly answered initialization sequence prompts.

1. The program writes the following message:

   'taskname' INITIALIZATION

2. The program then issues the following prompt:

```
DEFAULT CONFIGURATION=INTERNAL ;[CR,?,S,M,I]?
```

Responding with the Return key causes the program to assign the specified default configuration (internal).

Responding with a question mark causes the program to write the following help message and reissue the prompt:

```
*-------------------------------------------------*
*              C O N F I G U R A T I O N          *
*-------------------------------------------------*

CODE:      ACRONYM FOR
-------------------------------------
 S  :   SLAVE IN REMOTE MODE
 M  :   MASTER IN REMOTE MODE
 I  :   INTERNAL IN LOCAL MODE
-------------------------------------
```

Responding with S causes the program to test the controller as a slave in Remote mode. If a controller is configured as a slave in Remote mode, the program bypasses Tests 1 through 37, automatically runs Tests 38 through 55, and allows Tests 56 through 58 to be run individually by operator request.

Responding with M causes the program to test the controller as a master in Remote mode. If a controller is configured as a master in Remote mode, the program bypasses Tests 1 through 37, automatically runs Tests 38 through 55, and allows Tests 56 through 58 to be run individually by operator request.

Responding with I causes the program to test the controller as internal in Local mode. If a controller is configured as internal in Local mode, the program runs Tests 1 through 37 exclusively. Tests 38 through 58 cannot be run if a controller is configured as internal in Local mode.

3. The program then issues the following prompt:

```
DEFAULT CHANNEL ADDRESS=0C ;[CR,##]?
```

Responding with the Return key causes the program to assign the specified default channel address (0C). To modify the default channel address, enter the valid two-character hexadecimal channel address corresponding to the configuration.

4. The program then issues the following prompt:

   DEFAULT SUBADDRESS(ES) TO TEST=0,8;[CR,#(,#)]?

   Responding with the Return key causes the program to assign the specified default subaddresses (0,8). To modify the default subaddresses, enter two to 16 hexadecimal subaddresses separated by commas.

   Note: The program requires that two subaddresses be specified containing a receive subchannel address which is any integer in the range 0 through 7 and a corresponding transmit subchannel address which is any integer in the range 8 through F.

5. The program then issues the following prompt:

   DEFAULT INDIVIDUAL ADDRESS=123456789ABC ;[CR,############]?

   Responding with the Return key causes the program to assign the specified default ethernet individual address (123456789ABC). To modify the default individual address, enter a one to 12 character hexadecimal address corresponding to the configuration.

   The program compares the destination field of each incoming frame to the specified ethernet individual address.

   Note: For Test 58, the ethernet individual address cannot be zero.

6. The program then issues the following prompt:

   # OF MULTICAST ADDR TO LOAD FOR SOURCE STATION=1 ;[CR,0,1,2,3]?

   Responding with the Return key causes the program to assign the specified default number of multicast addresses (1). To modify the default number of multicast addresses, enter the valid number of multicast addresses.

   If executing the tests provided by this program, the program uses only the first multicast address specified.

   Note: For Tests 33 and 54, specify at least one multicast address.

7. The program then issues the following prompt:

```
1) DEFAULT MULTICAST ADDRESS=010000000123 ;[CR,############]?
```

Responding with the Return key causes the program to assign the specified default multicast address (010000000123). To modify the default multicast address, enter a valid multicast address. If more than one multicast address is specified, the program issues a prompt for each multicast address.

Note: Set the least significant bit (bit 7) of the most significant byte (byte 0) of all multicast addresses.

8. The program then issues the following prompt:

```
DEFAULT DESTINATION ADDRESS=123456789ABC ;[CR,############]?
```

Responding with the Return key causes the program to assign the specified default destination address (123456789ABC). To modify the default destination address, enter the one to 12-character hexadecimal destination address. The program inserts the destination address into the frame, identifying which controller is to receive a frame of data.

Note: Internal mode requires that the ethernet individual address be set equal to the destination address. If the individual address does not equal the destination address, the program reissues the previous prompt.

9. The program then issues the following prompt:

```
DEFAULT MULTICAST ADDR FOR DESTINATION STATION=010000000123
;[CR,############]?
```

Responding with the Return key causes the program to assign the specified default multicast address (010000000123). To modify the default multicast address, enter the valid one to 12 character hexadecimal multicast address.

Note: Set the least significant bit (bit 7) of the most significant byte (byte 0) of all multicast addresses.

10. The program then writes the following messages:

```
CHANNEL ** INITIALIZED
TO REPEAT THIS SEQUENCE:        DXP>REINIT 'taskname'
TO START THE TEST SEQUENCE:     DXP>START 'taskname'
    'taskname' INITIALIZATION COMPLETE
```

The first three messages are only for information. The fourth message indicates that initialization is successful. At this time, modify options and/or start the program.

2.3 Program Option Set

Once loaded and initialized, the Gould Ethernet Controller Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould Ethernet Controller Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T] ? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loop on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option causes the program to repeat the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resettng the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Unimplemented.
9	Reset	Reserved, do not use.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file.

Option	Default	Effect If Set
11	Set	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Inhibits error message expansion.
13	Reset	Inhibits status error messages.
14	Reset	Unimplemented.
15	Reset	Inhibits data error messages.
16	Reset	Expands bit definitions. Setting this option expands all status, sense, statistics, and queue bit definitions if actual equals expected data.
17	Reset	Enables Command Trace.
18	Reset	Enables Data Recorder.
19	Reset	Enables Data Recorder on error.
20	Reset	Inhibits data comparison.
21	Reset	Reserved, do not use.
22	Reset	Reserved, do not use.
23	Reset	Reserved, do not use.
24	Reset	Reserved, do not use.

CHAPTER 3

PROGRAM EXECUTION

3.1 Starting Program Execution

After setting options, issue a Diagnostic Executive Program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-988r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Milestone Messages

If Option 11 (writes milestone messages) is set, the program writes the following messages to the LO file prior to the execution of each test or subtest (as applicable):

```
hh:mm:ss 'taskname' TEST: test number  test description
hh:mm:ss 'taskname' SUBTEST: subtest number
```

If Option 10 (writes pass count messages) is set, the program writes the following message to the LO file after completing all designated tests:

```
hh:mm:ss 'taskname' FINISHED PASS: pass count
```

If Option 6 (repeats program) is reset, the program issues the following prompt to the OC file after completing all tests in a single pass:

```
'taskname'  WHAT NEXT?
```

The program suspends execution at this point and issues a DXP> prompt.

3.3 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Message Routines

The Gould Ethernet Controller Diagnostic verifies the correct operation of the controller (I/O command) function through a fixed sequence of checks. If the program detects an error and Option 0 (inhibits error messages) is reset, it writes a message describing actual and expected results during execution of an I/O command.

The program writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   <<<<'taskname'----hh:mm:ss>>>>

   taskname — specifies the one to eight character program name as indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   TST: ttt, DEV:aa-dsa, PC:ppppp

   ttt — specifies the decimal test number of the erring test.

   aa — specifies the two-character channel address.

   dsa — specifies the two-character device subaddress.

   ppppp — specifies the program counter value within the program.

3. The erring directive (for directive tests only)

   DRTV: directive

   directive — specifies the erring directive.

4. The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable)

```
CMND: xxxxxxxx, (cccccccc),IOCL ADDRESS: 00qqqqqq
```

xxxxxxxx	specifies the instruction issued.
cccccccc	specifies the hexadecimal value of the instruction.
qqqqqq	specifies the address of the IOCL queue if the Extended I/O instruction is a Start I/O instruction. Otherwise, the program omits the value.

5. The IOCL queue contents

```
IOCD(ii):  oobbbbbb     ff00cccc
```

ii	specifies the queue slot number of I/O command doubleword (IOCD) in the IOCL queue.
oo	specifies the contents of the order byte in IOCD Word 1.
bbbbbb	specifies the initial buffer address in IOCD Word 1.
ff	specifies the contents of the flag byte in IOCD Word 2.
cccc	specifies the hexadecimal value of the initial byte count.

4.1.1 STARTIO (Start Input/Output) Routine and Error Messages

The STARTIO routine initiates an operation to a channel/controller/device. If the program executes an I/O instruction, the CPU sets certain bits in the condition code register reflecting the channel's response to the instruction. The program checks these condition codes and writes an error message if an unexpected result occurs.

If an unexpected error occurs in the STARTIO routine, the program classifies it as a fatal error and terminates after reporting this error condition. After a fatal error condition occurs and the program terminates execution, the channel still may post status and service interrupts. Issue a Reset Controller directive prior to resuming testing. To issue a Reset Controller directive, enter:

```
DXP> taskname RSCTL
```

Issuing a Reset Controller directive resynchronizes the program. If the controller is not reset after a fatal error, subsequent test results may be inaccurate.

If the ethernet controller under test does not accept a previously transmitted I/O operation, the program writes the following message to the operator communications (OC) file:

```
I/O OPERATION NOT ACCEPTED BY CHANNEL
CONDITION CODE: nnnn
```

nnnn specifies the pattern of bits in the condition code register.

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro described in the Permanent Macro Recovery Routines section.

If a service interrupt occurs while an I/O operation is in progress, the program writes the following message:

```
STARTIO WARNING: DEVICE dsa
RESYNCHRONIZING SYSTEM FOR NEW I/O OPERATION PREVIOUS I/O OPERATION
CLOSED WITHOUT DIAGNOSIS
```

dsa specifies two-character hexadecimal device subaddress.

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro.

If a service interrupt is expected but is not received, the program writes the following message:

```
STARTIO WARNING: DEVICE dsa
PREVIOUS I/O OPERATION IS STILL OPEN
CONTINUING MAY CAUSE A SOFTWARE GENERATED CHANNEL REJECT
```

dsa specifies the two-character hexadecimal device subaddress.

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro.

If a service interrupt occurs which is not expected, the program writes the following message:

```
STARTIO WARNING : DEVICE dsa
RESYNCHRONIZING SYSTEM FOR NEW I/O OPERATION
UNEXPECTED SERVICE INTERRUPT WAS RECEIVED
```

dsa specifies the two-character hexadecimal device subaddress.

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro.

4.1.2 WAIT4SI (Wait for Service Interrupt) Routine and Error Messages

The WAIT4SI routine waits for completion of I/O instructions which begin in the STARTIO routine. After a channel/controller/device completes the I/O instruction, it interrupts the CPU and transfers two words of status to main memory. The program waits 5 seconds for each I/O instruction.

If a service interrupt does not occur within the five-second time frame, the program writes a message describing the error and issues a Halt I/O instruction to the channel/controller/device in error, clearing the device for subsequent testing. If the program fails to receive a service interrupt as a result of the Halt I/O instruction, it writes another error message, classifies it as a fatal error condition, and terminates execution. Otherwise, testing continues.

If a fatal error condition occurs, issue a Reset Controller directive to resynchronize the program.

If a service interrupt is not received within five seconds of an I/O operation, the programs writes the following message:

I/O TIMEOUT (NO SERVICE INTERRUPT)

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro described in the Permanent Macro Recovery Routines section.

If a service interrupt is not received within five seconds of a Halt I/O instruction, the program writes the following message:

I/O TIMEOUT (NO SERVICE INTERRUPT FROM HALT I/O)

Continue running the program. No recovery is required for this error.

If a service interrupt is received but not expected, the program writes the following message:

UNEXPECTED SI RECEIVED

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro.

If the I/O status address stored in the Interrupt Context Block does not match the location previously defined by the Initialize Channel command, the program writes the following message:

INCORRECT STATUS POINTER RECEIVED:
EXP:xxxxxxxx ACT:xxxxxxxx

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro.

4.1.3 COPYSTAT (Copy Status) Routine and Error Messages

If a service interrupt occurs in WAIT4SI, the program analyzes the status of the channel/controller/device. The two words of status from the WAIT4SI routine contain a general description of the results of I/O instructions which begin in the STARTIO routine. The program receives more specific information by issuing a Start I/O Sense command. The COPYSTAT routine compares the actual status to the expected status and reports any discrepancies. This routine also requests sense data after each I/O instruction, compares the actual sense data to the expected sense data, and reports any discrepancies.

The program uses status and sense masks only with the Halt I/O command. The program also uses a mask of the ending status address to determine at which point in an I/O Command List (IOCL) queue the execution terminates after issuing a Halt I/O command. Then the program predicts status and sense data dynamically from actual commands to the channel/controller/device.

Note: The program fails to detect illegal directive sequences. Therefore, monitor execution closely during operator-composed directive tests. The program reports errors only if the channel/controller/device operates incorrectly, so the program reads the illegal directive sequence data and attempts to predict and to match it with actual results.

If an error occurs in the COPYSTAT routine and Option 13 (inhibits status error messages) is reset, the program writes the following message to the listed output (LO) file:

```
CHANNEL STATUS ERROR(S):
EXP STS (1):ooaaaaaa   (2):bbccrrrr
ACT STS (1):ooaaaaaa   (2):bbccrrrr
```

EXP STS — indicates the expected status values.

ACT STS — indicates the actual status values received.

oo — specifies the controller and device address of the device presenting the status.

aaaaaa — specifies the address of the I/O command executed plus eight bytes.

bb — specifies the channel status.

cc — specifies the controller status.

rrrr — specifies the channel's remaining byte count.

If Option 12 (inhibits error message expansion) is reset, the program writes one of following messages for each status bit using a plus sign to indicate an unexpected status bit and a minus to indicate a missing status bit in either the actual or expected status reports.

If Option 16 (expands status/sense definitions) is reset, the program skips messages indicating it receives an expected status bit.

Status Bit	Message	Explanation
8...	NOT USED	
4...	PPCI	specifies a IOCD Post Program Controlled Interrupt flag.
2...	INCORRECT LENGTH	indicates that the number of bytes transferred from the incoming frame exceeds the specified byte transfer count in the IOCD.
1...	PROGRAM CHECK	specifies one of the following programming errors: . invalid IOCD address specification . invalid IOCD address . invalid command . invalid count . invalid data address . invalid sequence
.8..	DATA CHECK	specifies that the program detects a parity error in main memory.
.4..	CONTROL CHECK	specifies that the program detects a hardware error while fetching an IOCD from Host memory.
.2..	INTERFACE CHECK	specifies that the program detects an error while transferring network data to or from the Host.
.1..	NOT USED	
..8.	BUSY	specifies that the controller fails to execute the command or I/O instruction to an active subchannel.
..4.	NOT USED	
..2.	NOT USED	
..1.	NOT USED	
...8	CHANNEL END	specifies that the subchannel is available for use in another operation.
...4	DEVICE END	specifies that the subchannel is available for use in another operation.
...2	UNIT CHECK	specifies that the program detects one of the following conditions: . a programming error . a hardware error
...1	UNIT EXCEPTION	specifies that the controller fails to transmit a frame to the network before reaching the retry count.

4.1.4 BUFFCOMP (Buffer Compare) Routine and Error Messages

If the STARTIO routine issues an I/O instruction involving a transfer of data, the BUFFCOMP routine compares the actual data to the expected data and writes an error message describing any discrepancies. If the program attempts a data transfer and the channel/controller/device executes it incorrectly, the program reports no errors in the data.

Dump the buffers using the DEXP directives DI (dump input buffer) and DO (dump output buffer).

If the destination address does not equal the expected address and Option 15 (inhibits data error messages) is reset, the program writes the following message to the listed output (LO) file:

```
FRAME ADDRESS ERROR(S)
DESTINATION ADDRESS EXPECTED=aaaaaaaa
DESTINATION ADDRESS RECEIVED=bbbbbbbb
SOURCE ADDRESS RECEIVED=cccccccc
```

aaaaaaaa specifies the expected destination address in the IOCD.

bbbbbbbb specifies the actual destination address received for the IOCD.

cccccccc specifies the actual source address received for the IOCD.

```
DATA ERROR(S)
BY#:    EXP:   ACT:
ffff      dd     dd
xxxx OUT  OF   yyyy
```

BY# indicates the byte number in error.

EXP indicates the expected input buffer value.

ACT indicates the actual input buffer value.

ffff specifies the byte number in error relative to the first byte of the IOCD where the error occurs.

dd specifies the hexadecimal value of the expected or actual data byte.

xxxx specifies the total number of bytes in error for the IOCL excluding the header of each frame.

yyyy specifies the total number of bytes (header and data) compared for the IOCL.

The program verifies that the total number of frames equals the number of bytes specified in the initial byte count for each IOCD. For the terminal IOCD, if the remaining byte count status indicates an incomplete byte transfer, the program only compares the transferred bytes.

4.1.5 ENDERMSG (End Error Message) Routine and Error Messages

The ENDERMSG routine implements Option 2 (stops on error), Option 3 (loops on error), Option 4 (loops on subtest), and terminates an error sequence. If an I/O command reject or a time-out on a Halt I/O occurs, the program terminates execution.

If a global error occurs, the program issues the following prompt:

'taskname' SUSPEND: ERROR [C,H,T]?

Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

If the controller is not responding to an I/O instruction, the program writes the following message:

'taskname' TERMINATED: DEVICE INOPERABLE => PLEASE RESYNCHRONIZE!!

Issue a Reset Controller command. If the error is not cleared, use the appropriate reset controller macro described in the Permanent Macro Recovery Routines section.

If an error occurs while attempting to find the corresponding DCE (Device Control Block Extension) for the device subaddress, the program writes the following message:

'taskname' TERMINATED
DEV:dsa NOT SELECTED, OR ILLEGAL SUBADDRESS

dsa specifies a two-character hexadecimal device subaddress.

Reinitialize the program assigning a valid subaddress.

4.2 Unique Messages

In addition to the standard error messages, the program writes the following additional messages:

If in Test 58 an incoming frame without an appropriate destination address is received, the program writes the following message:

```
ILLEGAL FRAME RECEIVED
```

If the last IOCD contains a Transfer-in-Channel (TIC) back to IOCD 1, the program writes the following message:

```
WARNING: DEVICE dsa
CURRENT QUEUE IS IN AN ENDLESS LOOP
```

dsa specifies a two-character hexadecimal device subaddress

Issue a Stop I/O instruction.

If an unrecoverable error occurs during program testing, the program writes the following message:

```
'taskname' TERMINATED: PLEASE RESYNCHRONIZE => SLAVE AND MASTER
                       CONTROLLERS!!
```

Abort slave and/or master programs and restart all slave controllers prior to the master controller.

If an error occurs during execution of the CRST directive and Option 0 (inhibits error messages) is reset, the program writes the following message:

```
READ STATISTICS ERROR(S)
EXP STATISTICS (1):xxxxxxxx
ACT STATISTICS (1):xxxxxxxx
        .               .
        .               .
        .               .

EXP STATISTICS (7):xxxxxxxx
ACT STATISTICS (7):xxxxxxxx
```

xxxxxxxx specifies the statistics counter.

The action to be taken for this error depends on the error word (1 through 7) and the statistics counter.

The following is a list of the error words and associated statistics counters:

Error Word	Halfword	Statistics Count Definition
(1)	xxxx....	frame alignment error count.
	xxxx	frame CRC error count.
(2)	xxxx....	local BUS available error count.
	xxxx	local buffer overflow error count.
(3)	xxxx....	transmit collision count.
	xxxx	read frames with length error count (IEEE 802.3 only).
(4)	xxxx....	frames successfully written after more than one collision.
	xxxx	transmit deferred count.
(5)	xxxx....	frames unsuccessfully written.
	xxxx	frames successfully written after one collision.
(6)	xxxxxxxx	frames successfully read.
(7)	xxxxxxxx	frames successfully written.

If an error occurs during execution of the CSNS directive and Option 0 (inhibits error messages) is reset, the program writes the following message:

```
DEVICE SENSE ERROR(S):
DEVICE dsa
EXP SNS (1):ffffrrrr
ACT SNS (1):ffffrrrr
```

dsa specifies a two-character hexadecimal device subaddress.

ffff specifies the sense bits.

rrrr specifies the byte count of the previous frame.

The action to be taken for this error depends on the error specified in the sense bit. The following is a list of the sense bits and associated descriptions:

Sense Bit	Expansion	Sense Bit Definition
0	8...	command reject
1	4...	NA
2	2...	NA
3	1...	equipment check
4	.8..	NA
5	.4..	NA
6	.2..	mode field (2**1)
7	.1..	mode field (2**0)
8	..8.	receive unit ready
9	..4.	transmission deferred
10	..2.	collision field (2**1)
11	..1.	collision field (2**0)
12	...8	no SQE test flag
13	...4	underrun flag
14	...2	lost clear to send
15	...1	no carrier sense flag

Bits 6 and 7 together relate to the operating mode. Bits 10 and 11 together relate to the status of the current frame. The following are expansions of the mode bit and collision field sense bit definitions:

Bit 6	Bit 7	Mode	Bit 10	Bit 11	Frame Status
0	0	0 (Transparent)	0	0	Successful
0	1	1 (Ethernet 1.0)	0	1	Successful (1 pass)
1	0	2 (Ethernet 2.0)	1	0	Successful (2-15 passes)
1	1	3 (IEEE 802.3)	1	1	Unsuccessful (16 passes)

4.3 Permanent Macro Recovery Routines

If an error is generated which cannot be resolved by issuing a Reset Controller command, use the following two recovery routines:

Permanent Macro RSCL

If the controller is configured as internal in Local mode, use this macro to reset the channel, and initialize and configure the controller:

RST,RSCHL,DCI,ECI,INCH,CONF(INT),LIA(RMB),SNS

Permanent Macro RSCR

If the controller is configured as slave or master in Remote mode, use this macro to reset the channel, and initialize and configure the controller:

RST,RSCHL,DCI,ECI,INCH,CONF,LIA(RMB),SNS

4.4 Initialization Error Messages

If errors exist in the channel due to a configuration or hardware problem, the program writes the following messages to the operator communications (OC) file:

If the channel is inoperable or invalid, the program writes the following message:

'taskname' TERMINATED: ttaa-dsa ABSENT

tt specifies the assigned device type code.

aa specifies the channel address.

dsa specifies the two-character device subaddress.

The previous message indicates that the specified channel is not responding to commands.

If the specified channel cannot be initialized, the program writes the following message:

'taskname' TERMINATED: UNABLE TO INITIALIZE ttaa-dsa

tt specifies the assigned device type code.

aa specifies the channel address.

dsa specifies the device subaddress.

4.5 Run-Time Error Codes

If directive sequences are entered incorrectly, the program writes one of the following run-time error codes:

Code	Description
RT01	A directive references an undefined directive type.
RT02	A jump directive references an undefined statement number.
RT03	A directive specifies an undefined letter modifier.
RT04	A directive specifies an illegal count modifier.
RT05	A directive fails to convert an expected number string into its binary equivalent.
RT06	The program detects an illegal build queue string directive.
RT08	The program detects fatal error conditions that prevent further execution.
RTks	A literal directive specifies an undefined or illegal keyword (k) or subfield (s).

4.6 Queue-drive Error Codes

If an attempt is made to build I/O Command List (IOCL) queues which are illegal or undefined, the program writes one of the following queue-drive error codes:

Code	Description
QD01	The program attempts to build an IOCL queue of more than 15 entries.
QD02	The program detects an illogical queue sequence. This indicates serious problems with the program in memory.
QD03	The program attempts to build a read IOCL whose cumulative transfer count exceeds the defined buffer boundaries for that device.
QD04	The program specifies an illegal transfer-in-channel (TIC) address.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould Ethernet Controller (EC.ENT.r) contains a sequence of 58 tests which test the ethernet controller in one of three configurations: Internal in Local mode, Slave in Remote mode, or Master in Remote mode. The following is a list of the tests which execute automatically for each configuration:

Configuration	Tests Executed Automatically
Internal in Local mode	1 through 37
Slave in Remote mode	38 through 55
Master in Remote mode	38 through 55

In addition to the tests which execute automatically for each configuration, Tests 56 through 58 execute if the controller is configured as either a Slave or Master in Remote mode. These tests execute individually by operator request and involve a one way transmission with the master controller transmitting to each of the configured slave controllers. The master controller fails to wait for a response from the slave controller.

5.2 Test Execution

This program tests an ethernet controller both individually and as a part of a network. If testing a controller as a part of a network, the program must be in complete control of the network to generate accurate results. The network must consist of a series of point-to-point consecutive controllers in a ring shaped configuration with each controller on the network transmitting only to one other controller, without transmitting to the same controller.

In this configuration, controller one transmits to controller two, which transmits to controller three. The last controller in the ring transmits back to controller one. Establish this ring shaped network during initialization by using valid Ethernet Individual Addresses and unique destination addresses for each controller.

After the program loads, initializes, and starts, each slave controller on the network waits until receiving a transmission. The master controller transmits a frame to one slave controller. After receiving the frame, the slave transmits a frame to another slave controller. The program continues this process until the last slave controller in the ring receives a frame and in turn transmits a frame to the master controller.

The following are recommendations for testing the Ethernet Controller efficiently:

- Configure each controller as internal in Local mode first. This allows a single controller to be isolated for test purposes. The 37 tests executed during this pass cause the controller to perform increasingly difficult tasks.

- Once the controller successfully executes the first 37 tests, assume that the controller is running all internal functions correctly. Reinitialize each controller, except the controller reinitialized as the master controller, as a slave in Remote mode. The program causes the slave controller to send out a Read command and wait until receiving data.

  Note: System clock override must be OFF before executing Remote mode tests.

- Reinitialize the controller to be tested as the master, as master in Remote mode. The program causes the master controller to send out a Write command immediately followed by a Read command. At this point, the program begins executing Tests 38 through 55:

  Note: If testing two or more controllers in Remote mode, start the slave units before the master unit for synchronization.

5.3 Test 1 No Operation (NOP)

This directive test verifies that the ethernet controller initializes and executes a No Operation (NOP) instruction.

5.4 Test 2 Reset Controller and Sense Command

This directive test verifies that the controller correctly accepts and returns sense information upon receipt of a Sense command.

5.5 Test 3 Invalid Commands

This directive test verifies that the controller returns proper status if a Start I/O with an invalid command is issued.

5.6 Test 4 Program Check Test

This in-line coded test verifies proper status for all program check conditions. The test contains five subtests:

- Subtest 4A verifies that the controller detects and reports an invalid transfer-in-channel as the first word of an I/O command doubleword sequence (TIC-in-IOCD1).
- Subtest 4B verifies that the controller detects and reports an invalid transfer-in-channel to transfer-in-channel sequence (TIC-to-TIC).
- Subtest 4C verifies that the controller detects and reports an invalid address.
- Subtests 4D and 4E verify that the controller detects and reports an invalid count.

5.7 Test 5 Load Individual Address

This directive test verifies that the controller loads an Ethernet Individual Address to all active units, with the multicast bit set.

5.8 Test 6 Transfer-in-Channel

This directive test verifies that the controller correctly executes an I/O Command List (IOCL) queue containing multiple Transfer-in-Channel commands.

5.9 Test 7 No Operation With Post Program Controlled Interrupt (PPCI)

This directive test verifies that the controller correctly performs a Post Program Controlled Interrupt (PPCI) if executing command chained No Operation commands from an I/O Command List (IOCL) queue.

5.10 Test 8 Halt I/O (No I/O in Progress)

This directive test verifies that the controller correctly executes a Halt I/O command if no I/O is in progress.

5.11 Test 9 Read Statistics Data Chaining

This in-line coded test verifies that the controller correctly reads the statistics for currently configured channels using data chaining.

5.12 Test 10 Stop I/O

This in-line coded test verifies that the controller suppresses command chaining in the current IOCD, and terminates the operation after completion of the IOCD by executing a Stop I/O instruction.

5.13 Test 11 Incorrect Length

This in-line coded test verifies that the controller correctly responds if given a transfer count of incorrect length.

5.14 Test 12 Subchannel Busy

This in-line coded test verifies that the controller correctly responds if a specified subchannel is busy.

5.15 Test 13 Invalid Instruction

This in-line coded test verifies that the controller correctly responds if issued an invalid instruction.

5.16 Test 14 Sense Command With Skip Flag

This directive test verifies that the controller correctly executes a Sense command if the skip flag is set.

5.17 Test 15 Halt I/O (Read in Progress)

This in-line coded test verifies that the controller correctly executes a Halt I/O instruction while receiving data.

5.18 Test 16 Halt I/O (Write in Progress)

This directive test verifies that the controller correctly executes a Halt I/O instruction while transmitting data.

5.19 Test 17 Promiscuous Mode (Loopback)

This directive test verifies that the controller correctly transfers data if placed in a promiscuous mode using a single controller. Promiscuous mode is set if the multicast bit is set. If in promiscuous mode, a controller reads every valid frame regardless of the frame's destination address.

5.20 Test 18 Broadcast Address Recognition (Loopback)

This directive test verifies that the controller correctly transmits and receives data with a broadcast address using a single controller.

5.21 Test 19 Read Statistics and Sense (Loopback)

This in-line coded test verifies that the controller correctly transmits and receives data, returning the correct sense and status information, using a single controller.

5.22 Test 20 Varying Transfer Count (Loopback)

This directive test verifies that the controller correctly sends and receives transmissions with differing even numbered transfer counts using a single controller.

5.23 Test 21 Read Data Chaining (Loopback)

This directive test verifies that the controller correctly receives data using data chaining on a single controller.

5.24 Test 22 Write Data Chaining (Loopback)

This directive test verifies that the controller correctly transmits data using data chaining on a single controller.

5.25 Test 23 Read and Write Data Chaining (Loopback)

This directive test verifies that the controller correctly performs data chained Read and Write commands using a single controller.

5.26 Test 24 Write Data Chained With Byte Addresses (Loopback)

This directive test verifies that the controller correctly transmits and receives data using data chained writes with byte addresses on a single controller.

5.27 Test 25 Read with Invalid Address (Loopback)

This in-line coded test verifies that a single controller correctly detects and reports an attempt to receive data from an invalid address.

5.28 Test 26 Read and Write Command Chaining (Loopback)

This directive test verifies that the controller performs command chained Read and Write commands using a single controller.

5.29 Test 27 Transfer-in-Channel (Loopback)

This directive test verifies that the controller correctly performs a Transfer-in-Channel (TIC) command if transmitting or receiving data using a single controller.

5.30 Test 28 Post Program Controlled Interrupt (Loopback)

This directive test verifies that the controller correctly receives data with the Post Program Controlled Interrupt (PPCI) flag set using a single controller.

5.31 Test 29 Read Variable Length Frames From Local Storage (Loopback)

This directive test verifies that the controller correctly receives previously transmitted data from local storage using a varying even numbered count.

5.32 Test 30 Read Multiple Frames From Local Storage (Loopback)

This directive test verifies that the controller buffers 16 incoming data frames and correctly returns the data if 16 Read commands are issued using a single controller.

5.33 Test 31 Long Frame Status (Loopback)

This directive test verifies that the controller correctly transmits a frame which is longer than the count for the received frame using a single controller.

5.34 Test 32 Short Frame Status (Loopback)

This directive test verifies that the controller correctly transmits a frame which is shorter than the count for the received frame using a single controller.

5.35 Test 33 Multicast Address Recognition and Disable Multicast Frames (Loopback)

This directive test verifies that the controller correctly transmits and receives data both with and without a multicast address using a single controller.

5.36 Test 34 Read With Skip Flag (Loopback)

This directive test verifies that the controller correctly executes a Read command with the skip flag set using a single controller.

5.37 Test 35 Transmit and Receive Data on the Same Subchannel (Loopback)

This in-line coded test verifies that the controller correctly transmits and receives data using a single subchannel on a single controller.

5.38 Test 36 Local Buffer Overflow Error (Loopback)

This directive test verifies that the controller correctly recognizes and reports a local buffer overflow using a single controller.

5.39 Test 37 Nonpresent Memory Access

This in-line coded test verifies that the controller correctly detects an invalid memory access condition if the request is to transfer data into a nonpresent memory module.

Note: The program skips this test automatically on systems with 16MB of memory.

5.40 Test 38 Ping Pong Mode

This directive test verifies that the controller correctly alternates transmit and receive requests between the master controller and all configured slave controllers.

5.41 Test 39 Varying Transfer Counts

This directive test verifies that the controller correctly transmits and receives data to another Ethernet controller using both even and odd differing transfer counts.

5.42 Test 40 Read Data Chaining

This directive test verifies that the controller correctly receives data from another Ethernet controller using data chaining.

5.43 Test 41 Write Data Chaining

This directive test verifies that the controller correctly transmits data to another Ethernet controller using data chaining.

5.44 Test 42 Read and Write Data Chaining

This directive test verifies that the controller correctly transmits and receives data from another Ethernet controller using data chaining.

5.45 Test 43 Read Command Chaining

This directive test verifies that the controller correctly receives data from another Ethernet controller using command chaining.

5.46 Test 44 Write Command Chaining

This directive test verifies that the controller correctly transmits data to another Ethernet controller using command chaining.

5.47 Test 45 Read and Write Command Chaining

This directive test verifies that the controller correctly receives and transmits data to another Ethernet controller using command chaining.

5.48 Test 46 Transfer-in-Channel (TIC)

This directive test verifies that the controller correctly performs a Transfer-in-Channel (TIC) command if either transmitting or receiving data from another Ethernet controller.

5.49 Test 47 Post Program Controlled Interrupt Flag Set

This directive test verifies that the controller correctly receives data from another Ethernet controller with the Post Program Controlled Interrupt (PPCI) flag set.

5.50 Test 48 Read Variable Length Frames From Local Storage

This directive test verifies that the controller correctly transmits data to and receives data from another Ethernet controller using even and odd differing transfer counts. This test also verifies the type-ahead feature if previously transmitted data is correctly received.

5.51 Test 49 Read Multiple Frames From Local Storage

This directive test verifies that the controller correctly receives multiple records from previously transmitted data residing in local storage.

5.52 Test 50 Long Frame Status

This directive test verifies that the controller correctly transmits a frame which is longer than the count for the received frame using two or more controllers.

5.53 Test 51 Short Frame Status

This directive test verifies that the controller correctly transmits a frame which is shorter than the count for the received frame using two or more controllers.

5.54 Test 52 Multicast Address Recognition and Disable Multicast Address

This directive test verifies that the controller correctly transmits and receives data both with and without a multicast address using two or more controllers.

5.55 Test 53 Read with Skip Flag

This directive test verifies that the controller correctly executes a Read command from another Ethernet controller with the skip flag set.

5.56 Test 54 Local Buffer Overflow

This test verifies that the controller correctly identifies if a local buffer overflow occurs using two or more controllers.

5.57 Test 55 Contention

This directive test verifies that the controller correctly transfers data if there is contention on the network.

5.58 Test 56 Promiscuous Mode

This directive test verifies that the controller correctly transfers data if in Promiscuous mode. Promiscuous mode is set if the multicast bit is set. If in Promiscuous mode, a controller reads every valid frame regardless of its destination address.

This test is a one-way transmission with the master controller transmitting to each of the configured slave controllers.

5.59 Test 57 Broadcast Address Recognition

This directive test verifies that the controller correctly transmits and receives data with a broadcast address. This is a one-way transmission with the master controller transmitting to each of the configured slave controllers.

5.60 Test 58 Destination Address Recognition Frame Reject

This in-line coded test verifies that the controller correctly rejects incoming data which has no appropriate destination address using two or more controllers.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould Ethernet Controller Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould Ethernet Controller Diagnostic parameters and a summary of their functions follow:

Parameter	Type	Description
A	Reserved	Contains the flags used by the Assign processor.
B	Available	Available for general use.
C	Reserved	Contains invalid command byte for SIO.
D	Available	Available for general use.
E	Reserved	Contains the Power Up flag.
F	Reserved	Contains the infinite time-out flag for the Read commands.
G	Reserved	Contains the incorrect length flag.
H	Reserved	Contains the default mode.
I	Available	Available for general use.
J	Reserved	Contains the slave/master designator.
K	Reserved	Contains the flag value.
L	Reserved	Contains the configuration type code.
M	Reserved	Contains the mode field default.
N	Reserved	For internal use only.
O	Available	Available for general use.
P	Reserved	Contains the maximum legal transfer count for a Write directive.
Q	Reserved	Contains the default count to use if no absolute transfer count is specified. The Write directive uses this parameter.
R	Reserved	Contains Buffer Input file definition table address.
S	Reserved	Contains the current subtest designator (in binary).
T	Available	Available for general use.
U	Reserved	Contains the minimum legal transfer count.
V	Reserved	Contains the maximum transfer count allowed for a Read directive.
W	Reserved	Contains the default count to use if no absolute transfer count is specified. The Read directive uses this parameter as the transfer count default value.
X	Reserved	Reserved for internal use.
Y	Reserved	Reserved for internal use.
Z	Reserved	Contains the indirect address for the JZ directive.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in paramter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

The following is a list of valid string subfields used with the BQ, BRQ, and BWQ directives:

CC	Sets the command chaining flag in the I/O command doubleword (IOCD).
DC	Sets the data chaining flag in the IOCD.
PPCI	Sets the post program controlled interrupt flag in the IOCD.
SIL	Sets the suppress incorrect length indication flag in the IOCD.
SKIP	Sets the skip flag in the IOCD.
N	Sets the byte count for the IOCD.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L=' 'or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD(op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PI(op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PZ= $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould Ethernet Controller Diagnostic Unique Directives

The Gould Ethernet Controller Diagnostic contains unique directives which perform functions specific to the program.

The Gould Ethernet Controller Diagnostic unique directives follow.

6.6.1 "A" Directive

AC
AC(U#)

Issues an Activate Channel Interrupt instruction. If a unit number is specified (U#), the program issues the Activate Channel Interrupt instruction only to the specified unit.

6.6.2 "B" Directives

BQ(arg_1,...,arg_n)

Builds a chained I/O Command List (IOCL) queue which contains the specified arguments for all active subchannels. The maximum allowed length for the IOCL queue is 15 entries. If no I/O command flags are specified by the arguments, the program sets data chaining.

The following Ethernet directives are used as arguments with the BQ directive: CCNTR, CONF, DMF, IVD, LIA, LMA, NOP, R, RFTM, RSTAT, SNS, TIC, and W.

Display this IOCL queue by issuing the DCQ directive.

BRQ(arg_1,...,arg_n)

Builds a chained read IOCL queue which contains the specified arguments for all active receive subchannels. The maximum allowed length for the IOCL queue is 15 entries. If no I/O command flags are specified by the arguments, the program sets data chaining.

The following Ethernet directives are used as arguments with the BRQ directive: CCNTR, CONF, DMF, IVD, LIA, LMA, NOP, R, RFTM, RSTAT, SNS, and TIC.

Display this IOCL queue by issuing the DCQ directive.

BWQ(arg_1,...,arg_n)

Builds a chained write IOCL queue that contains the specified arguments for all active transmit subchannels. The maximum allowed length for the IOCL queue is 15 entries. If no I/O command flags are specified within the arguments, the program sets data chaining.

The following Ethernet directives are used as arguments with the BWQ directive: CCNTR, CONF, DMF, IVD, LIA, LMA, NOP, RFTM, RSTAT, SNS, TIC, and W.

Display this IOCL queue by issuing the DCQ directive.

6.6.3 "C" Directives

CCNTR
CCNTR(U_1,...,U_n)

Issues a Start I/O Clear Counter command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues a Start I/O Clear Counter command only to specified units.

CONF(arg_1,...,arg_n)
CONF(U_1,...,U_n,arg_1,...,arg_n)

Issues a Start I/O (Configure "LCC") command to execute specified arguments for all active units. If unit numbers are specified (U_1,...,U_n), the program executes the arguments only for the specified units. Use the following arguments with this directive:

Argument	Default	Operation If Set
BAD	Reset	Save bad frames.
BRD	Reset	Disable broadcast frames.
CR	Reset	Transmit on no carrier sense.
CRC	Reset	No CRC insertion.
EPRT(n)	n=0	Exponential priority. The range for "n" is 1 through 7.
EXT	Reset	External loop-back.
FIFO(n)	n=12	First-In-First-Out limit. The range for "n" is 1 through 15.
INT	Reset	Internal loop-back.
LGTH(nn)	nn=60	Minimum frame length. The range for "nn" is 1 through 255.
LPRT(n)	n=0	Linear priority. The range for "n" is 0 through 7 (zero is the highest priority).

CRST

Compares expected statistics to actual statistics. If expected does not equal actual, the program dumps both sets of statistics to the listed output (LO) file.

CSNS
CSNS(U_1,...,U_n)

Compares expected sense data to actual sense data for all active units. If expected does not equal actual, the program dumps both sets of sense data to the LO file. If unit numbers are specified (U_1,...,U_n), the program compares only sense data for the specified units.

6.6.4 "D" Directives

DA
DA(U#)

Issues a Deactivate Channel Interrupt instruction. If a unit number is specified (U#), the program issues the Deactivate Channel Interrupt instruction only to the specified unit.

DAS
DAS(U_1,...,U_n)

Dumps the actual status, sense data, and statistics of all active units to the listed output (LO) file. If unit numbers are specified (U_1,...,U_n), the program dumps only data for the selected units.

The program writes a header above the DAS table in the following format:

```
DEV: dsa
ACT STS (1):ooaaaaaa (2):bbccxxxx
ACT SNS (1):ffffrrrr
ACT STATISTICS (1):tttttttt
        .               .
        .               .
        .               .

ACT STATISTICS (7):tttttttt
```

dsa	specifies the two-character hexadecimal device subaddress.
oo	specifies the controller and device address of the device presenting status.
aaaaaa	specifies the address of the I/O command executed plus eight bytes.
bb	specifies the channel status.
cc	specifies the controller status.
xxxx	specifies the channel's remaining byte count.
ffff	specifies the sense bits.
rrrr	specifies the byte count of the previous frame.
tttttttt	specifies the actual statistics word value in the input buffer.

DCB
DCB(U_1,...,U_n)

Dumps the device control block (DCB) of all active units to the listed output (LO) file. If unit numbers are specified (U_1,...,U_n), the program dumps only data for the specified units. If the unit is inactive, the program writes the following message to the operator communications (OC) file:

VOID RANGE

The program writes a header above the DCB table in the following format:

DEV	FLAG	NOT USED	CMND	IOCLQP	IOTT	DCE ADDR	ACTUAL STATUS
00	:00000000	00000000	00000000	00000000	00000000	00000000	00000000+00000000

DEVC	identifies the subchannel number of the DCB displayed.
FLAG	identifies the DCB flag word.
NOT USED	indicates a place holder for future modification which is not currently in use.
CMND	identifies the last I/O command executed on this subchannel.
IOCLQP	identifies the I/O Command List queue address of the last executed command.
IOTT	identifies the I/O time-out system elapsed time value.
DCE ADDR	identifies the associated device control block extension (DCE) main memory address.
ACTUAL STATUS	identifies the actual status returned for the last I/O command executed on this subchannel.

DCE
DCE(U_1,...,U_n)

Dumps the device control block extension of all active units to the listed output (LO:) file. If unit numbers are specified (U_1,...,U_n), the program dumps only data for the specified units. If the unit is inactive, the program writes the following message to the operator communications (OC) file:

VOID RANGE

The program writes a header above the DCE table in the following format:

```
DEVC FLAG DCB ADDR OUT BUFF  IN BUFF  NOT USED  ASENSEP  EXPECTED STATUS
00 :00000000 00000000  00000000  00000000  00000000  00000000  00000000+00000000
```

DEVC	identifies the subchannel number of the device control block extension (DCE) displayed.
FLAG	identifies the device control block extension (DCE) flag word.
DCB ADDR	identifies the device control block (DCB) address.
OUT BUFF	identifies the associated output buffer address.
IN BUFF	identifies the associated input buffer address.
NOT USED	indicates a place holder for future modification which is not currently in use.
ASENSEP	identifies the actual sense buffer's main memory address.
EXPECTED STATUS	identifies the latest predicted status for this subchannel.

DCI
DCI(U#)

Issues a Disable Channel Interrupt instruction. If a unit number is specified (U#), the program issues the Disable Channel Interrupt instruction only to the specified channel.

DCQ
DCQ(U_1,...,U_n)

Dumps the current I/O Command List (IOCL) queue of all active units to the listed output (LO) file. If unit numbers are specified (U_1,...,U_n), the program dumps only data for the specified units. The program writes the following header and the contents of the current queue:

```
CURRENT QUEUE CONTENTS: dsa
```

dsa — specifies two-character hexadecimal device subaddress.

If the queue is empty, the program writes the following message to the operator communications (OC) file:

```
IOCL QUEUE EMPTY
```

If the device subaddress is inactive, the program writes the following message to the OC file:

```
VOID RANGE
```

DES
DES(U_1,...,U_n)

Dumps the expected status, sense, and statistics of all active units to the listed output (LO) file. If unit numbers are specified (U_1,...,U_n), the program dumps only data for the specified units.

The program writes a header above the DES table in the following format:

```
DEV: dsa
EXP STS (1):ooaaaaaa  (2):bbccxxxx
EXP SNS (1):ffffrrrr
EXP STATISTICS (1):tttttttt
        .                .
        .                .
        .                .
EXP STATISTICS (7):tttttttt
```

dsa — specifies the two-character hexadecimal device subaddress.

oo — specifies the expected controller and device address of the device presenting status.

aaaaaa — specifies the expected address of the I/O command executed plus eight bytes.

bb — specifies the expected channel status.

cc — specifies the expected controller status.

xxxx — specifies the channel's remaining byte count.

ffff — specifies the expected sense bits.

rrrr — specifies the byte count of the previous frame.

tttttttt — specifies the actual statistics word value in the input buffer.

DMF
DMF(U_1,...,U_n)

Issues a Start I/O Disable Multicast Frames command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Start I/O Disable Multicast Frames command only to specified units.

6.6.5 "E" Directive

ECI
ECI(U#)

Issues an Enable Channel Interrupt instruction. If a unit number is specified (U#), the program issues the Enable Channel Interrupt instruction only to the specified unit.

6.6.6 "H" Directives

HIO
HIO(U_1,...,U_n)

Issues a Halt I/O command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Halt I/O command only to the specified units.

HLP

Lists all directives which are specific to this program.

6.6.7 "I" Directives

INCH

Performs an Initialize Channel command.

IVD

Increments the invalid command byte in parameter C to the next invalid order and issues that invalid command to all active units.

6.6.8 "L" Directives

LADD
LADD(ONES)
LADD(arg_1,...,arg_n)

Loads destination and source addresses in first 12 bytes of the output buffer if mode is 0. If mode is 1, 2, or 3, this directive loads the destination address in the first six bytes of the output buffer. The jumper configuration determines the mode (refer to the Ethernet Controller Hardware Technical Manual for a description of jumper configurations).

Specify the destination, multicast, and source addresses during initialization and on each occurence of the Reinitialize command.

If ONES is specified, the program loads the first 12 bytes of the output buffer with all ones destination and source addresses if mode is 0. If mode is 1, 2, or 3 and ONES is specified, the program loads the first six bytes of the output buffer with an all ones destination address.

If arguments are specified (arg_1,...,arg_n) and mode is 0, the program loads the specified multicast and source addresses in the first 12 bytes of the output buffer. If mode is 1, 2, or 3, the program loads the specified multicast address in the first six bytes of the output buffer. The arguments, inserted into the frame by the program, identify which controller is to receive the frame of data.

The following are valid arguments:

M1	First multicast address specified at initial program load (IPL) and on each occurrence of the Reinitialize command.
M2	Second multicast address specified at IPL and on each occurrence of the Reinitialize command.
M3	Third multicast address specified at IPL and on each occurrence of the Reinitialize command.

LIA
LIA(arg_1,...,arg_n)
LIA(U_1,...,U_n,arg_1,...,arg_n)

Loads the ethernet individual address to all active units. If unit numbers are specified (U_1,...,U_n), the program loads the ethernet individual address only to specified units.

The following are valid arguments:

SMB	Load ethernet individual address with the multicast bit set.
RMB	Load ethernet individual address with the multicast bit reset.

LMA
LMA(U_1,...,U_n)

Loads the multicast addresses for the source station to all active units. The multicast addresses for source stations are specified at initialization and on each occurrence of the reinitialize command. If unit numbers are specified (U_1,...,U_n), the program loads the multicast addresses only to specified units.

6.6.9 "M" Directives

MIB(address)

Moves the input buffer to the specified address.

MIB(PGM)

Moves the input buffer to the default location specified by the Diagnostic Executive Program following program load.

MOB(address)

Moves the output buffer to the specified address.

MOB(PGM)

Moves the output buffer to the default location specified by the Diagnostic Executive Program following program load.

MRST(h##,d####)
MRST(U_1,...,U_n,h##,d####)

Modifies the halfword data specified by "d" in the expected statistics buffer halfword location indicated by "h" for all active units. If unit numbers are specified (U_1,...,U_n), the program modifies the halfword data specified by "d" in the expected statistics buffer halfword location indicated by "h" only for specified units. The program modifies as many as 14 halfwords with a single MRST directive.

MSNS(h#,d####)
MSNS(U_1,...,U_n,h#,d####)

Modifies the halfword data specified by "d" in the expected sense buffer halfword location indicated by "h" for all active units. If unit numbers are specified (U_1,...,U_n), the program modifies the halfword data specified by "d" in the expected sense buffer halfword location indicated by "h" only for specified units. The program modifies as many as two halfwords with a single MSNS directive.

6.6.10 "N" Directive

NOP
NOP(U_1,...,U_n)

Issues a NOP (No Operation) command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the NOP command only to the specified units.

6.6.11 "R" Directives

R
Rn
R(U_1,...,U_n)

Issues a Start I/O Read command of n bytes or, if n is not specified, the program reads parameter W bytes and issues the Read command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Read command only to the specified units.

RSCHL

Issues a Reset Channel instruction. The program writes the following message to the listed output (LO) file after issuing the RSCHL instruction:

PLEASE WAIT: RSCHL IN PROGRESS

RSCTL
RSCTL(U_1,...,U_n)

Issues a Reset Controller instruction to all units. If unit numbers are specified (U_1,...,U_n), the program issues the Reset Controller instruction only to the specified units. This directive also initializes the program to a reset condition for subsequent testing.

RST

Resets the program to initial conditions.

RSTAT
RSTAT(U_1,...,U_n)

Issues a Start I/O Read Statistics command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Start I/O Read Statistics command only to the specified units.

6.6.12 "S" Directives

SCM

Selects Concurrent I/O mode. In Concurrent mode, the program fails to wait for a subaddress to complete an I/O operation before executing an I/O operation on another subaddress.

SEL(ALL)
SEL(U_1,...,U_n)

Sets the Select flag in the device control block extension (DCE) for all active units. If unit numbers are specified (U_1,...,U_n), the program sets the select flag only on the specified units.

SIO

Issues a Start I/O and then waits for Service Interrupt (SI), Check Status data, and Data-If-Read input from the I/O Command List (IOCL) queue pointed to by specified commands in the current IOCL queue.

SNS
SNS(U_1,...,U_n)

Issues a Start I/O Sense command to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Start I/O Sense command only to the specified units.

SRT

Issues a Start I/O to receiver then transmitter and then waits for Service Interrupt (SI), Check Status data, and Data-If-Read input from the I/O Command List (IOCL) queue pointed to by specified commands in the current IOCL queue.

SSM

Selects Sequential I/O mode. In Sequential mode, the program waits for a subaddress to complete an I/O operation before executing an I/O operation on another subaddress.

ST

Establishes a subtest designator and, if Option 11 (writes milestone messages) is set, writes the subtest milestone message to the listed output (LO) file. The subtest designator value is in parameter S.

STOP
STOP(U_1,...,U_n)

Issues a Stop I/O instruction to all active units. If unit numbers are specified (U_1,...,U_n), the program issues the Stop I/O instruction only to the specified units.

6.6.13 "T" Directives

TICn

Inserts a Transfer-in-Channel (TIC) I/O command doubleword (IOCD) into the I/O Command List (IOCL) queue. The program inserts a TIC at the IOCD position specified by n. The value of n is 1 through 16 (decimal).

A TIC is an implied command chain operation as the flag byte of the previous IOCD passes through the TIC.

A TIC is never the first or last IOCD in the IOCL queue.

TIO
TIO(U#)

Issues a Test I/O instruction. If a unit number is specified (U#), the program issues the Test I/O instruction only to the specified unit.

6.6.14 "W" Directives

W
Wn
$W(U_1,...,U_n)$

Issues a Start I/O Write command of n bytes, or if n is not specified, the program writes parameter Q bytes. The program issues a Write command to all active units. If unit numbers are specified $(U_1,...,U_n)$, the program issues a Start I/O Write command only to the specified units.

6.6.15 "Z" Directives

ZRST

Zeros the expected Read Statistics buffer.

ZSNS
$ZSNS(U_1,...,U_n)$

Zeros expected sense buffer for all active units. If unit numbers are specified $(U_1,...,U_n)$, the program zeros only the specified units.

CHAPTER 7

I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview

The Gould Ethernet Controller Diagnostic contains optional capabilities that provide comprehensive listings of I/O activity during program execution.

These capabilities include I/O Command Trace and Data Recorder.

By setting the appropriate options, these routines produce a visual representation to the listed output (LO) file of the data being manipulated and I/O commands being issued.

Use these listings to examine in greater detail, the processor activity by supplementing error messages and verifying that the actual data being generated coincides with the commands being issued.

7.2 I/O Command Trace

The I/O Command Trace provides a record of all I/O instructions issued and in the order the program issues them during execution. Option 17 enables the I/O Command Trace and sends the information to the LO file.

7.2.1 Starting Execution

To begin I/O Command Trace execution, set Option 17 (enables Command Trace) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the LO file:

```
<<<<'taskname'----hh:mm:ss>>>>
TST:ttt, DEV:aa-dsa, PC:ppppp
DRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal channel address.
dsa	specifies the two-character hexadecimal device subaddress.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.2.2 Milestone Messages

The program writes the following milestone messages to the listed output (LO) file during an I/O Command Trace:

The I/O command issued and, if the command is a Start I/O, the I/O Command List (IOCL) queue pointer:

CMND:xxxxxxxx (cccccccc) ,IOCL ADDRESS:00qqqqqq

xxxxxxxx specifies the instruction issued.

cccccccc specifies the hexadecimal value of the instruction.

qqqqqq specifies the address of the IOCL queue.

The IOCL queue contents (repeated for all IOCDs within the IOCL queue):

If Option 16 (expands bit definitions) is reset, the program writes the following information:

	WORD 1	WORD 2	ORDER FUNCTION
IOCD(ii):	00bbbbbb	ff00cccc	order

ii specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo specifies the contents of the order byte in IOCD Word 1.

bbbbbb specifies the hexadecimal initial buffer address in IOCD Word 1.

ff specifies the contents of the flag byte contained in IOCD Word 2.

cccc specifies the hexadecimal value of the initial byte count in IOCD Word 2.

order specifies the IOCD order. The program writes a text description of the order as follows:

Order Byte Number	Order Function
00	INITIALIZE
01	WRITE
02	READ
03	NO OPERATION
04	SENSE
07	LOAD IND ADDR
08	T.I.C.
0B	DISABLE MULTICST
0F	LOAD MULTICAST
10	CONFIGURE
14	READ STATISTICS
15	CLEAR COUNTER
FF	INITIALIZE / NOP

If Option 16 (expands bit definitions) is set, the program writes the following information to the operator communications (OC) file:

	WORD 1	WORD 2	ORDER FUNCTION
IOCD(ii)	:oobbbbbb	ff00cccc	order
		:ff	flag
		:....xxxx	TRANSFER COUNT

ii specifies the queue slot number of the I/O command doubleword (IOCD) in the I/O Command List (IOCL) queue.

oo specifies the contents of the order byte in IOCD Word 1.

bbbbbb specifies the hexadecimal initial buffer address in IOCD Word 1.

ff specifies the contents of the flag byte in IOCD Word 2.

Following is a description of the flag byte values:

ff	Value
8.	DATA CHAIN
4.	COMMAND CHAIN
2.	SUPPRESS INCORRECT LENGTH
1.	SKIP
.8	POST PROGRAM CONTROLLED INTERRUPT

cccc specifies the hexadecimal value of the initial byte count in IOCD Word 2.

order specifies the IOCD order. If the listed output (LO) file assignment is to a console device, the program writes the hexadecimal value of the order byte. Otherwise, it writes a text description of the order.

flag specifies the IOCD flag description.

....xxxx specifies the decimal transfer count.

7.2.3 Terminating the I/O Command Trace

To terminate the I/O Command Trace, issue an Attention command and reset Option 17 (enables Command Trace) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 17
```

7.3 Data Recorder

The Data Recorder provides a dump of the actual data being manipulated during program execution. The dump shows how the data appears during I/O activity. With either Option 18 (enables Data Recorder) or Option 19 (enables Data Recorder on error) set, the routine sends a dump of data to the listed output (LO) file.

7.3.1 Starting Execution

To begin Data Recorder execution, set Option 18 (enables Data Recorder) or Option 19 (enables Data Recorder on error) using the DEXP Set Option command. During execution, the program writes the following start message, task separation banner, and header to the listed output (LO) file:

```
***START DATA RECORDER***
    <<<<'taskname'----hh:mm:ss>>>>
    TST:ttt, DEV:aa-dsa, PC:ppppp
    DRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal channel address.
dsa	specifies the two-character hexadecimal device subaddress.
ppppp	specifies the program counter value in the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.3.2 Milestone Messages

The program writes the following milestone messages during the Data Recorder routine:

If Option 16 (expands bit definitions) is reset, the program writes the following information to the listed output (LO) file:

```
    ACT STS    (1):ooaaaaaa        (2):bbccrrrr
```

oo	specifies the unit number.
aaaaaa	specifies the address of the I/O command executed plus eight bytes.
bb	specifies the channel status.
cc	specifies the controller status.
rrrr	specifies the remaining byte count.

If Option 16 (expands bit definitions) is set, the program writes the following information to the listed output (LO) file:

```
ACT STS  (1):ooaaaaaa (2):bbccrrrr
                          :xxxx      msg
```

oo	specifies the unit number.
aaaaaa	specifies the address of the I/O command executed plus eight bytes.
bb	specifies the channel status.
cc	specifies the controller status.
rrrr	specifies the remaining byte count.
xxxx	specifies the hexadecimal value of the status bit expansion message.
msg	specifies the status message.

Refer to the COPYSTAT routine for status bit descriptions.

7.3.3 Terminating Data Recorder

To terminate the Data Recorder routine, issue an Attention command and reset Option 18 (enables Data Recorder) or Option 19 (enables Data Recorder on error) using the DEXP Reset command.

Example (if Option 18 is set)

```
@@A
DXP>RESET taskname,OPT 18
```

Gould GPDC Asynchronous Line Interface Module

(GP.ALM.r)

Diagnostic Description

Diagnostic Number: 001032-986r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005740-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould GPDC Asynchronous Line Interface Module (GP.ALM.r) Diagnostic Description Release 1.0, Publication Order Number 326-005740-000, was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vii/viii
1-1/1-2
2-1 through 2-6
3-1 through 3-2
4-1 through 4-4
5-1 through 5-14
6-1 through 6-24
7-1 through 7-6

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.1.1 GPMC Model Number Assignments 2-2
2.1.2 Terminal Model Number Assignments 2-3
2.2 Wraparound Data Set Connection 2-3
2.3 Program Option Set 2-5

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Milestone Messages 3-1
3.3 Terminating Program Execution 3-2

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Message Routine 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Link Connect/Disconnect Verification 5-2
5.3 Test 2 Illegal Order 5-2
5.4 Test 3 Transmit and Receive Zeros 5-3
5.5 Test 4 Transmit and Receive Ones 5-3
5.6 Test 5 Transmit and Receive Floating Ones 5-3
5.7 Test 6 Transmit and Receive Floating Zeros 5-3
5.8 Test 7 Transmit and Receive Binary Progression 5-3
5.9 Test 8 Transmit and Receive Random Data 5-3
5.10 Test 9 Transmit and Receive Alternating Pattern: AA55AA55 5-4
5.11 Test 10 Transmit and Receive Varying Record Lengths 5-4
5.12 Test 11 Character Termination (Non-blind Mode) 5-4
5.13 Test 12 Character Recognition (STX-ETX) 5-4
5.14 Test 13 Character Termination (Blind Mode) Before ETX 5-5
5.15 Test 14 Character Termination (Blind Mode) After ETX 5-5
5.16 Test 15 Strip Character Recognition 5-5
5.17 Test 16 Ring Detection 5-5
5.18 Test 17 Ring Character Recognition 5-5

5.19 Test 18 Break Detection 5-6
5.20 Test 19 Parity Error Detection 5-6
5.21 Test 20 Framing Error Detection 5-6
5.22 Test 21 Transmit/Receive at Varying Baud Rates 5-6
5.23 Test 22 Transmit/Receive at Varying Character Lengths 5-7
5.24 Test 23 Transmit/Receive with No, Odd, Even Parity 5-7
5.25 Test 24 Transmit/Receive with Varying Stop Bits 5-7
5.26 Test 25 Residual Byte Count Status Verification 5-7
5.27 Test 26 Character Output (Alphabet) 5-8
5.28 Test 27 Character Output (Numerals) 5-8
5.29 Test 28 Character Output (Special Characters Group One) 5-8
5.30 Test 29 Character Output (Special Characters Group Two) 5-9
5.31 Test 30 Character Input (Alphabet) 5-9
5.32 Test 31 Character Input (Numerals) 5-9
5.33 Test 32 Character Input (Special Characters) 5-10
5.34 Test 33 Character Input (Control Characters) 5-10
5.35 Test 34 Character Recognition (STX-ETX) 5-10
5.36 Test 35 Single Character Termination 5-11
5.37 Test 36 Multiple Character Termination 5-12
5.38 Test 37 Strip Character Recognition 5-13
5.39 Test 38 Ring Detection 5-13
5.40 Test 39 Break Detection 5-14
5.41 Test 40 Framing Error Detection 5-14
5.42 Test 41 Parity Error Detection 5-14

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-4
6.6 Gould GPDC Asynchronous Line Interface Module Diagnostic Unique Directives 6-7
6.6.1 "A" Directives 6-7
6.6.2 "B" Directives 6-8
6.6.3 "C" Directives 6-9
6.6.4 "D" Directives 6-10
6.6.5 "E" Directives 6-12
6.6.6 "F" Directive 6-13
6.6.7 "H" Directives 6-13
6.6.8 "I" Directives 6-13
6.6.9 "L" Directives 6-14
6.6.10 "M" Directive 6-14
6.6.11 "R" Directives 6-15
6.6.12 "S" Directives 6-16
6.6.13 "T" Directives 6-19
6.6.14 "W" Directives 6-20
6.7 Control Information 6-21
6.8 Control Information Block (CIB) Directives 6-22
6.9 Control Information Block (CIB) Directive Arguments 6-22

CHAPTER 7 I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview 7-1
7.2 I/O Command Trace 7-1
7.2.1 Starting Execution 7-1
7.2.2 Milestone Messages 7-2
7.2.3 Terminating I/O Command Trace 7-2
7.3 Data Recorder 7-3
7.3.1 Starting Execution 7-3
7.3.2 Milestone Messages 7-3
7.3.3 Terminating Data Recorder 7-6

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould GPDC Asynchronous Line Interface Module Diagnostic (GP.ALM.r) is a level two macrodiagnostic program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 41 tests, which verify the correct operation of GPDC instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring operator intervention by setting appropriate options.

During execution, the program writes milestone messages and error message sequences to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The operator reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode). The CC file assignments are to disc or tape and OC file assignments are to the operator's console or a terminal configured on an eight-line asynchronous controller.

1.2 Resource Requirements

The Gould GPDC Asynchronous Line Interface Module Diagnostic requires:

- A Gould CPU with a minimum of 64K bytes of memory
- A GPDC ALIM Controller (Model 9110)
- A GPDC IOM/Controller (Model 9103, 16 megabyte), or
  A GPDC IOM/Controller (Model 9104, 512 kilobyte)

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould GPDC Asynchronous Line Interface Module (ALIM) Diagnostic is capable of testing from one to 16 asynchronous channels connected in various configurations. However, it tests only one type of configuration at a time.

The Gould ALIM Diagnostic is a self-initializing program and sets up its default values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the appropriate Diagnostic Facility User's Manual. The program name for loading the Gould ALIM Diagnostic is GP.ALM.r.

The Gould ALIM Diagnostic sets up the following default assignments:

- A jumpered Half-Duplex configuration
- A GPMC Controller address of 20
- GPDC ALIM channel address specifying channel 00_{16} jumpered to channel 01, and channel 02 jumpered to channel 03

To modify the default configuration, the GPMC Controller address, and/or the GPDC ALIM channel addresses, use the DEXP Assign command with the following device type codes:

Code	Description
TH	Terminal Half-Duplex
TF	Terminal Full-Duplex
JH	Jumpered Half-Duplex
JF	Jumpered Full-Duplex
MH	Modem Half-Duplex
MF	Modem Full-Duplex

In Full-Duplex mode, specify only the even-numbered channel of the Full-Duplex pair. In a jumpered or modem configuration, adjacent channel number pairs specify the channel connection for wraparound tests.

Use the directives AR, AT, ASR, and AST to assign the primary receive and transmit channels and also the secondary receive and transmit channels, respectively. The following table defines the channel numbers assigned by these directives for each of the configurations.

Device Type Code	Primary Receive (AR)	Primary Transmit (AT)	Secondary Receive (ASR)	Secondary Transfer (AST)
TH	C_1	C_0	C_0	C_0
TF	C_0	$C_0 + 1$	C_0	$C_0 + 1$
JH	C_0	C_1	C_0	C_1
JF	C_0	C_1	$C_1 - 1$	$C_0 + 1$
MH	C_0	C_1	C_0	C_1
MF	C_0	C_1	$C_1 - 1$	$C_0 + 1$

C_0 specifies the primary receive channel under test.

C_1 specifies the primary transmit channel connected to the primary receive channel for wraparound tests.

2.1.1 GPMC Model Number Assignments

The Gould GPDC Asynchronous Line Interface Module (ALIM) Diagnostic sets up the default model number assignment as 9103 (a GPMC IOM/Controller capable of addressing 16 megabytes of memory). To modify the assignment to Model 9104 (a GPMC IOM/Controller capable of addressing 512K bytes of memory, use the MOD directive.

2.1.2 Terminal Model Number Assignments

The Gould GPDC Asynchronous Line Interface Module (ALIM) Diagnostic sets up a Model 9204 ADM-3 terminal if executing in the Terminal Half-Duplex (TH) or Terminal Full-Duplex (TF) modes. To modify the terminal model number assignment, use the MOD directive and enter Model 9203 for a configured Hazeltine terminal.

The following are default operational parameters configured for each terminal:

Model 9203 Hazeltine	Model 9204 ADM-3
. 1 Stop bit	. $1\frac{1}{2}$ Stop bits
. Even Parity	. Odd Parity
. 7 Bits/Character	. 8 Bits/Character
. 16X Baud Rate Factor	. 16X Baud Rate Factor

2.2 Wraparound Data Set Connection

The following procedure describes the connection process required to operate the Gould GPDC Asynchronous Line Interface Module (ALIM) Diagnostic in a wraparound connection through Model 202S-L1A Data Sets. Consult the manufacturer's instructions with other data sets.

1. Assign the ALIM channels to be tested using the DEXP Assign command.

2. Set the baud rate, private/switched, and any other parameters required in the default control information block.

3. Enter the directive sequence:

   ```
   DXP>taskname AR,CCH,AT,CCH
   ```

   The program then writes the following message to the operator communications (OC) file:

   ```
   'taskname' SUSPENDED: CONNECT DEV:20-0
   ```

4. At the data sets, press the appropriate line buttons for the channels to be tested.

   Then, on the data set for the transmitting channel (DEV:20-1), dial the number of the receiving channel (DEV:20-0).

   After the phone rings at the receiving data set, lift the handset and press the DATA button.

   Ensure that the MR (Modem Ready) lamp lights after three seconds.

5. Enter the DEXP Continue command.

   The program writes the following message to the OC file:

   ```
   'taskname' SUSPENDED: CONNECT DEV:20-1
   ```

6. At the data set for the transmitting channel, press the Data button and ensure that the MR lamp lights after three seconds. Replace the handsets at both data sets.

7. Enter the DEXP Continue command.

   The program writes the following message to the operator communications (OC) file:

   ```
   'taskname' WHAT NEXT??
   ```

   At this time, modify options and/or start the program.

   If the link is disconnected during testing, the program writes an error message to the listed output (LO) file with a test device (TDV) status of Link Disconnected/Data Set Not Ready, Unusual End, and Channel End. The program terminates execution and writes the following message to the OC file:

   ```
   'taskname' TERMINATED: DEVICE INOPERABLE
   ```

   Reestablish the connection before resuming testing.

   To disconnect the link after testing is complete, lift the handsets at the data sets and press the line buttons, or enter the following directive string:

   ```
   DXP>taskname AR,DCH,AT,DCH
   ```

2.3 Program Option Set

Once loaded, the Gould GPDC Asynchronous Line Interface Module Diagnostic enters a wait state.

Using the DEXP, Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould GPDC Asynchronous Line Interface Module Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or any message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Inhibits operator intervention tests. Setting this option inhibits any test requiring operator intervention.

Option	Default	Effect If Set
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count message to the listed output (LO) file.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone message to the listed output (LO) file.
12	Reset	Inhibits error message expansion.
13	Reset	Inhibits test device status error messages.
14	Reset	Inhibits test buffer address status error messages.
15	Reset	Inhibits data error messages.
16	Reset	Inhibits test command list address error messages.
17	Reset	Enables Command Trace.
18	Reset	Enables Data Recorder.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a DEXP Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-986r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Milestone Messages

The program verifies the device assignments entered and if the device type code assigned is other than TH,TF,JH,JF,MH, or MF, the program writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED:  INVALID DEVICE TYPE CODE tt
```

Reassign the device to be tested using one of the valid device type codes.

The program then verifies the presence of the GPMC IOM/Controller and, if it is absent, it writes the following message to the OC file:

```
'taskname' TERMINATED:  GPMC:aa ABSENT
```

Reassign the device to be tested using the correct GPMC IOM/Controller address.

The program then verifies that only even GPDC Controller addresses are assigned in a Full-Duplex mode. If an odd GPDC controller address is specified, the program writes the following message to the OC file:

```
'taskname' TERMINATED:  GPDC ASSIGNMENT ERROR
```

Reassign the channels to be tested specifying only even GPDC Controller addresses.

The program then verifies the presence of Asynchronous Line Interface Module Controllers (ALIM) and, if they are absent, the program writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED:  GPDC:c ABSENT
```

Reassign the device to be tested using the correct GPDC ALIM Controller address.

The program then sequentially executes all designated tests for each assigned GPDC Controller.

If Option 11 (writes milestone messages) is set, the program writes the following message to the listed output (LO) file prior to the execution of each test:

```
hh:mm:ss 'taskname' TEST n (test description)
```

If Option 10 (writes pass count messages) is set, the program writes the following message to the LO file after completing all designated tests:

```
hh:mm:ss 'taskname' FINISHED PASS: n
```

3.3 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Message Routine

The Gould GPDC Asynchronous Line Interface Module Diagnostic verifies the correct operation of the controller (I/O command) function through a fixed sequence of checks. If the program detects an error and Option 0 (inhibits error messages) is reset, it writes a message describing actual and expected results during execution of an I/O command.

The program writes all error messages to the listed output (LO) file which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<<'taskname '----hh:mm:ss>>>>
   ```

   taskname — specifies the one to eight character program name indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TST:ttt,DEV:aa-c-00,PC:ppppp
   ```

   ttt — specifies the test number of the erring test.

   aa — specifies the two-character hexadecimal GPMC Controller address.

   c — specifies the one-character hexadecimal ALIM channel address.

   ppppp — specifies the program counter value within the program.

3. The erring directive (for directive tests only)

   ```
   DRTV:directive
   ```

   directive — specifies the erring directive.

4. The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable)

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx — specifies the Command Device (CD) instruction issued.

qqqqqq — specifies the address of the IOCL queue if the CD instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the value.

5. The IOCL queue contents

```
IOCD(ii):ooffcccc  00bbbbbb
```

ii — specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo — specifies the contents of the order byte in IOCD Word 1.

ff — specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb — specifies the initial buffer addresses in IOCD Word 2.

4.2 Unique Error Messages

```
STATUS ERROR
EXP(DV):ddddcccc  (BA):00bbbbbb  (CLA):00aaaaaa
ACT(DV):ddddcccc  (BA):00bbbbbb  (CLA):00aaaaaa
```

ii — specifies the queue slot number of the IOCD in the IOCL queue where the status error occurred.

EXP — identifies the expected status values.

ACT — identifies the actual status values.

dddd — specifies the hexadecimal value of the test device (TDV) device status.

cccc — specifies the hexadecimal value of the TDV remaining byte count.

bbbbbb — specifies the hexadecimal value of the test buffer address (TBA) status.

aaaaaa — specifies the hexadecimal value of the test command list address (TCLA) status.

If Option 12 (inhibits error message expansion) is reset, the program writes one of the following messages for each status bit using a plus sign to indicate an unexpected status bit and a minus sign to indicate a missing status bit in either the actual or expected status reports.

```
8...    LINK DISCONNECTED/DATA SET NOT READY
4...    CARRIER OFF
2...    FRAMING ERROR
1...    RECEIVER OVERRUN
.8..    PARITY ERROR
.4..    RING DETECTED
.2..    BREAK DETECTED
.1..    CHANNEL INOP/NOT INITIALIZED
..8.    N/A
..4.    N/A
..2.    TRANSMISSION ERROR
..1.    INCORRECT LENGTH
...8    UNUSUAL END
...4    ILLEGAL ORDER
...2    INTERRUPT PENDING
...1    CHANNEL END
```

If the program executes a specified I/O instruction and receives no service interrupt (SI), it writes the following message:

```
I/O TIMEOUT (NO SI)
```

If data comparison discrepancies occur, the program writes the following message:

```
DATA ERROR(S)
FR#:    BFO:     BFI:
ffff    dd  ccc  dd  ccc
 .       .   .    .   .
 .       .   .    .   .
 .       .   .    .   .
ffff    dd  ccc  dd  ccc
xxxx OUT OF yyyy
```

ii — specifies the queue slot number of the IOCD in the IOCL queue where the data errors occurred.

FR# — identifies the frame numbers in error.

EXP — identifies the expected input buffer value.

ACT — identifies the actual input buffer value.

ffff — specifies the frame number in error relative to the first frame of the IOCD where the error occurred (a maximum of five error frames are reported for each IOCD).

dd — specifies the hexadecimal value of the expected or actual data element.

ccc — specifies the data byte as an inactive control character in the control information block (CIB) for the receive channel as follows:

ccc =	STX	Start-of-Text
	ETX	End-of-Text
	CHA	Termination Character A
	CHB	Termination Character B
	CHC	Termination Character C
	SC1	Strip Character 1
	SC2	Strip Character 2

xxxx — specifies the total number of frames in error for the IOCD.

yyyy — specifies the total number of frames compared for the IOCD.

The total number of frames compared by the program is equal to the number of bytes specified in the initial byte count for each IOCD. However, for the terminal IOCD, if the test device remaining byte count indicates that not all bytes requesting transfer are transferred, the program only compares those bytes actually transferred.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould GPDC Asynchronous Line Interface Module Diagnostic contains 41 tests. The program selects a subset of these tests depending on the device type code specified in the DEXP Assign command. Valid device type codes are as follows:

Device Type	Tests
TH	1,26 through 41
TF	1,26 through 41
JH	1 through 25
JF	1 through 17,19 through 25
MH	2 through 10,12,17 through 18
MF	2 through 10,12,17

The program executes the subset of the tests selected for a particular assignment mode in a continuous sequence, or executes each test individually by operator request.

Skip tests by selecting the appropriate options.

For test sequences run with device type codes MH or MF, transfer the control information to the device, issue a Connect Channel order, and establish the wraparound connection through the modems prior to starting test execution. Do this by entering the following directive string interactively through the command control (CC) file:

```
DXP > taskname AR,CCH,AT,CCH
```

At the execution of each CCH directive, the program writes the following message to the operator communications (OC) file and suspends execution to establish the connection:

```
'taskname' SUSPENDED: CONNECT DEV:aa-c
```

Continue program execution by entering the DEXP Continue command.

Set the default control information block to match the device parameters prior to executing tests with assignments of device type codes TH, TF, MH, and MF.

Set the device parameters using the DCI directive.

5.2 Test 1 Link Connect/Disconnect Verification

Execution Modes: JH, JF, TH, TF

This test contains two subtests which verify that the ALIM detects Device Inoperable and Link Disconnected conditions.

The program writes the following message to the operator communications (OC) file after suspending program execution:

'taskname' SUSPENDED: DISCONNECT JUMPER CABLE DEV:aa-c [C,H,T] ?

Disconnect the cable connected to the ALIM channel specified by GPMC address "aa" and GPDC address "c". Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

The program writes the following message to the OC file:

'taskname' SUSPENDED: RECONNECT JUMPER CABLE DEV:aa-c [C,H,T] ?

Reconnect the cable for the previously disconnected ALIM. Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

The program repeats this test for all assigned devices on the first pass of execution. The program inhibits the test until a new Assign command is issued.

The program skips this test if Option 7 (runs directive tests only) or Option 8 (inhibits operator intervention tests) is set.

5.3 Test 2 Illegal Order

Execution Modes: JH, MH, MF, JF

This test verifies that the GPDC Controller detects illegal IOCL orders and posts the correct test device (TDV) status for illegal orders.

The program skips this test if Option 7 (runs directive tests only) is set.

5.4 Test 3 Transmit And Receive Zeros

Execution Modes: JH, JF, MH, MF

This is a data transfer test used primarily to detect picked bits.

5.5 Test 4 Transmit and Receive Ones

Execution Modes: JH, JF, MH, MF

This is a data transfer test used primarily to detect dropped bits.

5.6 Test 5 Transmit and Receive Floating Ones

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transfers a one bit into each bit position of the data byte.

5.7 Test 6 Transmit and Receive Floating Zeros

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transfers a zero bit into each bit position of the data byte.

5.8 Test 7 Transmit and Receive Binary Progression

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transfers each eight-bit binary data pattern.

5.9 Test 8 Transmit and Receive Random Data

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transmits and receives a random data pattern.

5.10 Test 9 Transmit and Receive Alternating Pattern: AA55AA55...

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transmits and receives an alternating bit pattern.

5.11 Test 10 Transmit and Receive Varying Record Lengths

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM transmits and receives records of varying lengths.

5.12 Test 11 Character Termination (Non-blind Mode)

Execution Modes: JH, JF

This test verifies that the ALIM terminates a Non-Blind mode reception when it detects a character termination sequence. The test contains eight subtests that verify each of the five character termination codes, as defined in the following table:

Subtest	Termination Code	Code Function	Subtest Terminates On
11 A	1:	T(A)	A
11 B	2:	T(A+B)	A
11 C	2:	T(A+B)	B
11 D	3:	T(A+B+C)	A
11 E	3:	T(A+B+C)	B
11 F	3:	T(A+B+C)	C
11 G	4:	T(A.B)	AB
11 H	5:	T(A.B.C)	ABC

5.13 Test 12 Character Recognition (STX-ETX)

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM recognizes the Start-of-Text (STX) and End-of-Text (ETX) characters.

5.14 Test 13 Character Termination (Blind Mode) Before ETX

Execution Modes: JH, JF

This test verifies that the ALIM terminates a Blind mode reception after detecting a termination sequence while the receiver is not blind (the termination sequence occurs after Start-of-Text (STX) and prior to End-of-Text (ETX)).

The test contains eight subtests that verify the termination sequences for each of the five character termination codes, as defined in the following table:

Subtest	Termination Code	Code Function	Subtest Terminates On
13 A	1:	T(A)	A
13 B	2:	T(A+B)	A
13 C	2:	T(A+B)	B
13 D	3:	T(A+B+C)	A
13 E	3:	T(A+B+C)	B
13 F	3:	T(A+B+C)	C
13 G	4:	T(A.B)	AB
13 H	5:	T(A.B.C)	ABC

5.15 Test 14 Character Termination (Blind Mode) After ETX

Execution Modes: JH, JF

This test verifies that the ALIM terminates a blind mode reception after it detects a termination sequence while the receiver is blind (the termination sequence occurs after the End-of-Text (ETX) character is recognized).

The test contains eight subtests that verify the termination sequences for each of the five character termination codes, as defined in the following table:

Subtest	Termination Code	Code Function	Subtest Terminates On
14 A	1:	T(A)	A
14 B	2:	T(A+B)	A
14 C	2:	T(A+B)	B
14 D	3:	T(A+B+C)	A
14 E	3:	T(A+B+C)	B
14 F	3:	T(A+B+C)	C
14 G	4:	T(A.B)	AB
14 H	5:	T(A.B.C)	ABC

5.16 Test 15 Strip Character Recognition

Execution Modes: JH, JF

This test verifies that the ALIM deletes the characters specified as strip character one and strip character two.

This test is composed of three subtests:

- Subtest 15 A tests for deletion of strip character one.
- Subtest 15 B tests for deletion of strip character two.
- Subtest 15 C tests for deletion of both strip character one and strip character two.

5.17 Test 16 Ring Detection

Execution Modes: JH, JF

This test verifies that the ALIM detects a ring interrupt.

5.18 Test 17 Ring Character Recognition

Execution Modes: JH, JF, MH, MF

This test verifies that the ALIM recognizes the ring character (Character D) and generates a ring interrupt.

5.19 Test 18 Break Detection

Execution Modes: JH, MH

This test verifies that the ALIM detects Break as an attention function in Half-Duplex mode and as a transmission interrupt function in Full-Duplex mode.

5.20 Test 19 Parity Error Detection

Execution Modes: JH, JF

This test verifies that the ALIM detects and reports a parity error condition.

5.21 Test 20 Framing Error Detection

Execution Modes: JH, JF

This test verifies that the ALIM detects and reports a framing error condition.

5.22 Test 21 Transmit/Receive at Varying Baud Rates

Execution Modes: JH, JF

This directive test verifies that the ALIM transmits and receives at each of the selectable baud rates. This test contains 15 subtests, one for each baud rate as follows:

Subtest	Baud Rate
21A	50
21B	75
21C	110
21D	134.5
21E	150
21F	300
21G	600
21H	900
21I	1200
21J	1800
21K	2400
21L	3600
21M	4800
21N	7200
21O	9600

5.23 Test 22 Transmit/Receive at Varying Character Lengths

Execution Modes: JH, JF

This test verifies that the ALIM transmits and receives data at each of the selectable character lengths. It contains four subtests, one for each character length as follows:

Subtest	Character Length
22A	5 Bits
22B	6 Bits
22C	7 Bits
22D	8 Bits

5.24 Test 23 Transmit/Receive with No, Odd, Even Parity

Execution Modes: JH, JF

This test verifies that the ALIM transmits and receives with no parity, even parity, and odd parity. It contains three subtests, one for each parity mode as follows:

Subtest	Parity Selected
23A	No
23B	Even
23C	Odd

5.25 Test 24 Transmit/Receive with Varying Stop Bits

Execution Modes: JH, JF

This test verifies that the ALIM transmits and receives with one, one and a half, and two stop bits. It is composed of three subtests as follows:

Subtest	Number of Stop Bits
24A	1
24B	1 1/2
24C	2

5.26 Test 25 Residual Byte Count Status Verification

Execution Modes: JH, JF

This test verifies that the ALIM correctly sets the remaining byte count returned in the Test Device (TDV) status. It tests the remaining byte count for each bit position up to the maximum possible for the program's buffer size (i.e., for RBC=1,2,4,8,...,PV/2).

5.27 Test 26 Character Output (Alphabet)

Execution Modes: TH, TF

This test writes the alphabetic characters A through Z to the OC file as follows:

```
CHARACTER OUTPUT:
THE ALPHABET (A-Z)- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

This test requires visual verification for quality and correctness.

5.28 Test 27 Character Output (Numerals)

Execution Modes: TH, TF

This test writes the numeric characters 0 through 9 to the OC file as follows:

```
CHARACTER OUTPUT:
THE NUMBERS (ZERO-NINE) 0 1 2 3 4 5 6 7 8 9
```

This test requires visual verification for quality and correctness.

5.29 Test 28 Character Output (Special Characters Group One)

Execution Modes: TH, TF

This test writes the special characters to the OC file as follows:

```
CHARACTER OUTPUT:
THE SPECIAL CHARACTERS-
EXCLAMATION POINT          !
QUOTATION MARK             "
NUMBER SIGN                #
DOLLAR SIGN                $
PERCENT SIGN               %
AMPERSAND                  &
LEFT PARENTHESIS           (
RIGHT PARENTHESIS          )
ASTERISK                   *
PLUS SIGN                  +
COMMA                      ,
HYPHEN                     -
```

This test requires visual verification for quality and correctness.

5.30 Test 29 Character Output (Special Characters Group Two)

Execution Modes: TH, TF

This test writes the special characters to the OC file as follows:

```
CHARACTER OUTPUT:
THE SPECIAL CHARACTERS-
PERIOD                    .
SLASH                     /
COLON                     :
SEMICOLON                 ;
LESS THAN SIGN            <
EQUAL SIGN                =
GREATER THAN SIGN         >
QUESTION MARK             ?
AT SIGN                   @
LEFT BRACKET              [
RIGHT BRACKET             ]
INSERT MARK               ^
UNDERSCORE                _
```

This test requires visual verification for quality and correctness.

5.31 Test 30 Character Input (Alphabet)

Execution Modes: TH, TF

This test writes the following message to the OC file:

```
ALPHABET RECOGNITION: ENTER-

QWERTYUIOP ASDFGHJKL ZXCVBNM
```

This test requires the same message to be entered by the operator.

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.32 Test 31 Character Input (Numerals)

Execution Modes: TH, TF

This test writes the following message to the OC file:

```
NUMBER RECOGNITION: ENTER-
1234567890
```

This test requires the same message to be entered by the operator.

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.33 Test 32 Character Input (Special Characters)

Execution Modes: TH, TF

This test writes the following message to the OC file:

```
SPECIAL CHARACTER RECOGNITION: ENTER-
!"#$%&()*+,-./: <=> ?@[ ] ^ _
```

This test requires that the same message be entered by the operator.

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.34 Test 33 Character Input (Control Characters)

Execution Modes: TH, TF

This test writes the following message to the OC file:

```
CONTROL CHARACTER RECOGNITION:
WHILE HOLDING THE CONTROL KEY (CTRL) DOWN, ENTER-

ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

This test requires that the same message be entered by the operator.

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.35 Test 34 Character Recognition (STX-ETX)

Execution Modes: TH, TF

This test verifies that the ALIM recognizes Start-of-Text (STX) and End-of-Text (ETX) characters. The test writes the message:

```
STX, ETX RECOGNITION STX(B),ETX(D) : ENTER-
ABCDEABCD
```

This test requires the same message be entered by the operator. The test is successful if the input buffer contains the following values at the completion of the test:

STX	ETX	STX	ETX
0042	4344	000042	4344

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.36 Test 35 Single Character Termination

Execution Modes: TH, TF

This test verifies that the ALIM detects and terminates reception with the single-character termination modes. The test is composed of six subtests as follows:

Subtest	Terminate	Service Interrupt
35A	T(A)	SI(A)
35B	T(A+B)	SI(A)
35C	T(A+B)	SI(B)
35D	T(A+B+C)	SI(A)
35E	T(A+B+C)	SI(A)
35F	T(A+B+C)	SI(A)

The program writes the following message to the OC file:

CHARACTER TERMINATION:

Then, for each subtest, the program writes the following messages which require an operator response:

Subtest 35 A:

T(A),SI(A) ENTER-
A

Subtest 35 B:

T(A+B),SI(A) ENTER-
A

Subtest 35 C:

T(A+B),SI(B) ENTER-
B

Subtest 35 D:

T(A+B+C),SI(A) ENTER-
A

Subtest 35 E:

T(A+B+C),SI(B) ENTER-
B

Subtest 35 F:

T(A+B+C),SI(C) ENTER-
C

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.37 Test 36 Multiple Character Termination

Execution Modes: TH, TF

This test verifies that the ALIM detects and terminates reception with the multiple-character termination modes. The test is composed of two subtests as follows:

Subtest	Terminate	Service Interrupt
36A	T(AB)	SI(AB)
36B	T(ABC)	SI(ABC)

The program writes the following message to the OC file:

CHARACTER TERMINATION:

Then, for each subtest, the program writes the following messages which require an operator response:

Subtest 36 A:

T(AB),SI(AB) ENTER-
AB

Subtest 36 B:

T(ABC),SI(ABC) ENTER-
ABC

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.38 Test 37 Strip Character Recognition

Execution Modes: TH, TF

This test verifies that the ALIM recognizes and deletes the strip characters. The test is composed of three subtests as follows:

Subtest	Strip Characters
37A	S(A)
37B	S(B)
37C	S(A+B)

The program writes the following message to the OC file:

STRIP CHARACTER RECOGNITION:

Then, for each subtest, the program writes the following messages which require an operator response:

Subtest 37A:

S(A): ENTER-
ABCD

Subtest 37B:

S(B): ENTER-
ABCD

Subtest 37C:

S(A+B): ENTER-
ABCD

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.39 Test 38 Ring Detection

Execution Modes: TH, TF

This test verifies the ability to recognize a ring character and to generate a ring interrupt. The test writes the message:

RING DETECTION: DEPRESS- (ACK)/(CTRL-F)

Respond by entering an ACK character (Control-F).

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.40 Test 39 Break Detection

Execution Modes: TH, TF

This test verifies that the ALIM recognizes and reports a Break condition. In Half-Duplex mode, Break is an attention function. In Full-Duplex mode, it terminates a transmission.

The test writes the following message to the OC file:

BREAK DETECTION - DEPRESS BREAK KEY

Respond by depressing the Break key. In Full-Duplex mode, the program transmits a continuous message of 57 lines to allow time for response. The message terminates after the program detects the Break condition.

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.41 Test 40 Framing Error Detection

Execution Modes: TH, TF

This test verifies that the ALIM detects a framing error condition. The test writes the following message to the OC file:

FRAMING ERROR DETECTION: DEPRESS- (ACK)/(CTRL-F)

The program complements the character length field of the control information block (CIB) to force a framing error condition. Respond by entering an ACK character (Control-F).

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

5.42 Test 41 Parity Error Detection

Execution Modes: TH, TF

This test verifies that the ALIM detects a parity error condition. The test writes the following message to the OC file:

PARITY ERROR DETECTION: DEPRESS- (ACK)/(CTRL-F)

The program complements the parity selection bit of the control information block (CIB) to force a parity error condition. Respond to the message by entering an ACK character (Control-F).

The program skips this test if Option 8 (inhibits operator intervention tests) is set.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould GPDC Asynchronous Line Interface Module Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components.

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T] ?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname,MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould GPDC Asynchronous Line Interface Module Diagnostic parameters and a summary of their functions follow:

Parameter	Use	Description
A	Reserved	Contains the input/output buffer address attributes (F=1,C=00).
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer data pattern if the output buffer is set using a PD directive.
E	Available	Available for general use.
F	Reserved	Contains the program control flags. Bit 28=1: Loop on Error directive sequence active Bit 29=1: Wraparound mode Bit 30=1: Data Communications (modem) mode Bit 31=0: Half-Duplex mode Bit 31=1: Full-Duplex mode
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Reserved	Contains the test device status mask pattern: B00-15: TDV Device Status B16-31: TDV Remaining Byte Count (RBC)
N	Reserved	Contains the secondary transmit ALIM channel address.
O	Reserved	Contains the secondary receive ALIM channel address.
P	Reserved	Contains the primary transmit ALIM channel address.
Q	Reserved	Contains the primary receive ALIM channel address.
R	Reserved	Contains the default record count.
S	Reserved	Contains a number between 1 and 26 (decimal) corresponding to a the letter from A through Z. It is used by the ST directive as a subtest designator.

Parameter	Use	Description
T	Available	Available for general use.
U	Available	Available for general use.
V	Reserved	Contains the default transfer count (in bytes).
W	Reserved	Contains the default transfer count for W and R directives.
X	Reserved	Contains the hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for JZ directives.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

$$J \begin{Bmatrix} n \\ Z \end{Bmatrix} /k$$

$$J \begin{Bmatrix} n \\ Z \end{Bmatrix} /p$$

$$J \begin{Bmatrix} n \\ Z \end{Bmatrix} /p(op)k$$

$$J \begin{Bmatrix} n \\ Z \end{Bmatrix} /p_1(op)p_2$$

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L='' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the next random number pattern.
LOIn	Loads the output buffer with the contents of the input buffer.
LOn	Loads the output buffer with all ones.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter directives perform as stated the following syntax:

Pp(op)k

$Pp_1(op)Pp_2$

PD(op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PI(op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PZ= $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(o)	specifies a legal parameter operands.
k	specifies a number (0 through 32,767).

"S" Directive

STPn — Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME
TIME(message) — Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould GPDC Asynchronous Line Interface Module Diagnostic Unique Directives

The Gould GPDC Asynchronous Line Interface Module Diagnostic contains unique directives which perform functions specific to the program.

The Gould GPDC Asynchronous Line Interface Module Diagnostic unique directives follow.

6.6.1 "A" Directives

AL='string'

Appends the literal string to the program's output buffer at the position parameter W plus 1. The program adds the length of the literal string to parameter W.

AR

Stores the ALIM channel address of the primary receive channel in the program's device definition table (DDT) and creates the I/O Command List (IOCL) necessary to communicate with the unit under test.

ASR

Stores the ALIM channel address of the secondary receive channel in the program's DDT and creates the IOCL necessary to communicate with the unit under test.

AST

Stores the ALIM channel address of the secondary transmit channel in the program's DDT and creates the IOCL necessary to communicate with the unit under test.

AT

Stores the ALIM channel address of the primary transmit channel in the program's DDT and creates the IOCL necessary to communicate with the unit under test.

6.6.2 "B" Directives

BFI='address'

Relocates the program's input buffer to the specified address. The address is any address within the 16 megabytes of memory if a model 9103 GPMC is configured, or any address within 512KB of memory if a model 9104 GPMC is configured. The length of the relocated buffer is identical to the default program buffer length.

BFI='PGM'

Relocates the program's input buffer back to the default location within the program.

BFO='address'

Relocates the program's output buffer to the specified address. The address is any address within 16 megabytes of memory if a model 9103 GPMC is configured, or anywhere within 512KB of memory if a model 9104 GPMC is configured. The length of the relocated buffer is identical to the default program buffer length.

BFO='PGM'

Relocates the program's output buffer to the default location within the program.

BRK

Issues a Start I/O instruction to transmit a Break order ($0B_{16}$) to the currently assigned ALIM channel.

6.6.3 "C" Directives

CCH

Issues a Start I/O instruction to transmit a Connect Channel order (03_{16}) to the currently assigned ALIM channel.

If the device type code assigned is MH or MF, the program writes the following message and suspends program execution to establish the connection through the modem:

'taskname' SUSPENDED: CONNECT DEV: aa-u

Continue program execution using the DEXP Continue command.

CDn

Sets the Code byte of the control information block (CIB) for the currently assigned ALIM channel to the value specified by n.

The following are valid values for n:

n	Description
0	No action
1	Terminate on Character A
2	Terminate on Character A or B
3	Terminate on Character A or B or C
4	Terminate on Character A followed by B
5	Terminate on Character A followed by B followed by C

CHA='nn'

Sets the Terminate on Character A byte in the CIB for the currently assigned ALIM channel to the hexadecimal value specified by nn.

CHB='nn'

Sets the Terminate on Character B byte in the CIB for the currently assigned ALIM channel to the hexadecimal value specified by nn.

CHC='nn'

Sets the Terminate on Character C byte in the CIB for the currently assigned ALIM channel to the hexadecimal value specified by nn.

CHD='nn'

Sets the Ring on Character D byte in the CIB for the currently assigned ALIM channel to the hexadecimal value specified by nn.

CIB

Dumps the contents of the CIB for the currently assigned ALIM channel to the listed output (LO) file.

6.6.4 "D" Directives

DBD

Disables break detection by resetting the Break Detection bit in the control information block (CIB) for the currently assigned ALIM channel.

DCB

Dumps the contents of the device control block (DCB) to the listed output (LO) file.

DCB='arg'

Dumps the contents of the element specified by the argument in the DCB to the LO file.

The following are valid arguments for the DCB and DCE directives:

Argument	Description
FLAG	Flag word
TLIS	Test Last Interrupt Status (this cell is used only in the DCB for GPDC 0 for each GPMC)
CMND	Last Start I/O instruction executed
CLQP	IOCL Queue Pointer
DCEP	DCB Extension Pointer
TDVS	Test Device Status
TCLA	Test Command List Address Status
TBAS	Test Buffer Address Status
CIB1	Control Information Block bytes 0 through 3
CIB2	Control Information Block bytes 4 through 7
CIB3	Control Information Block bytes 8 through 11
CIB4	Control Information Block byte 12
XTDV	Expected Test Device Status
XTCL	Expected Test Command List Address Status
XTBA	Expected Test Buffer Address Status
IOST	I/O Start System Elapsed Time
WTSI	Wait for Service Interrupt return address for no wait I/O operation

DCI='nn_1,...,nn_n'

Sets the elements of the default control information block (CIB) to the values specified by nn.

Null values of nn cause no change in the corresponding bytes of the default CIB, and trailing null values need not be specified.

Example

DCI='7A,,C8'

This example sets the first element (mode byte) to $7A_{16}$ and the third element (format byte) to $C8_{16}$. The second element (rate byte) is not changed.

The following are preset values for the default CIB elements:

Mode	= 93_{16}		One Stop Bit, Even Parity, Eight bit Character Length, 16X Baud Rate Factor (ADM-3 compatible)
Rate	= $0F_{16}$		9600 Baud Rate
FRMT	= $C0_{16}$	*	Ring Detection Enabled
	or $D0_{16}$	*	Break Detection Enabled, 1ms R25 Off Delay Enabled, Half- or Full-Duplex depending on configuration), Private Line
CODE	= 00_{16}	*	No action
STX	= 10_{16}	*	Start-of-Text
ETX	= 30_{16}	*	End-of-Text
CHA	= 41_{16}	*	Terminate Character A
CHB	= 42_{16}	*	Terminate Character B
CHC	= 43_{16}	*	Terminate Character C
CS1	= 80_{16}	*	Strip Character One Inhibited
CS2	= 80_{16}	*	Strip Character Two Inhibited

*These fields should not be changed if executing the standard tests.

DRD

Disables ring detection by resetting the Ring Detection bit in the CIB for the currently assigned ALIM channel.

DQ

Dumps the contents of the I/O Command List (IOCL) queue for the currently assigned channel to the listed output (LO) file in the following format:

```
TST:ttt, DEV:aa-u-00, PC:ppppp
DRTV:DQ
IOCD(ii):ooffcccc 00bbbbbb
   .          .          .
   .          .          .
   .          .          .
```

ttt — specifies either the test number containing the DQ directive, or specifies KBD (keyboard directive) if entered from the command control (CC) file.

aa — specifies the two-character hexadecimal GPMC Controller address.

u — specifies the one-character hexadecimal ALIM channel address.

ppppp — specifies the hexadecimal program counter value in the DQ directive.

ii — specifies the queue slot number of the IOCD in the IOCL queue.

oo — specifies the contents of the order byte in IOCD Word 1.

ff — specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb — specifies the hexadecimal value of the initial buffer address in IOCD Word 2.

6.6.5 "E" Directives

EBD

Enables break detection by setting the Break Detection bit in the control information block (CIB) for the currently assigned ALIM channel.

ERD

Enables ring detection by setting the Ring Detection bit in the CIB for the currently assigned ALIM channel.

ETX='nn'

Establishes the End-of-Text (ETX) character in the CIB for the currently assigned ALIM channel as the value specified by nn.

6.6.6 "F" Directive

FDX

Sets Full-Duplex mode by setting the Full- or Half-Duplex bit in the control information block (CIB) for the currently assigned ALIM channel.

6.6.7 "H" Directives

HDX

Sets Half-Duplex mode by resetting the Full- or Half-Duplex bit in the CIB for the currently assigned ALIM channel.

HIO

Issues a Halt I/O instruction to the currently assigned ALIM channel.

6.6.8 "I" Directives

IBR

Increments the baud rate byte of the currently assigned ALIM channel by one. If the resulting baud rate exceeds $0F_{16}$ (9600 baud), the program resets the baud rate to a 01_{16} (50 baud).

IOR

Issues a I/O Reset instruction to the currently assigned ALIM channel.

CAUTION

An I/O Reset instruction causes the program to generate a master clear for all GPDC controllers configured on the assigned GPMC. Therefore, executing an I/O Reset in one GPDC program can cause errors to occur in other GPDC programs executing concurrently.

6.6.9 "L" Directives

LPE

Establishes a loop on error option for operator-composed tests. If Option 3 (loops on error) is set, this directive establishes a Loop on Error return point for all detected errors until the program issues another LPE or LPEND directive.

This directive also establishes a region of No-wait I/O for Read and Write directive operations to allow concurrent Read and Write operations for wraparound tests.

Within an LPE, LPEND pair, the program checks I/O completion and status by using the WIO directive.

Example

The directive sequence LPE,AR,R4,AT,W4,WIO,AR,WIO,LPEND transmits four bytes on the primary transmit channel and receives four bytes on the primary receive channel. Then the program checks the completion and status of each channel.

Initiate the Read operation prior to the Write operation for wraparound tests.

LPEND

Establishes the normal single directive Loop on Error mode as defined by Option 3 (loops on error) and causes the program to wait for I/O completion on Read and Write operations.

6.6.10 "M" Directive

MODn

Aligns all internal parameters consistent with the model number defined by n.

The following are valid model numbers (n) to use with this directive:

n	Parameters Used By Program
9103	16 megabyte GPMC parameters
9104	512 kilobyte GPMC parameters
9202	KSR-43 terminal parameters (TF or TH modes only)
9203	Hazeltine terminal parameters (TF or TH modes only)
9204	ADM-3 terminal parameters (TF or TH modes only)

6.6.11 "R" Directives

R
Rn

Constructs an I/O Command List (IOCL) Read queue and issues a Start I/O instruction to initiate a Read operation to the currently assigned ALIM channel.

Construction of the IOCL queue is data driven and is based on the contents of the control information block (CIB) for the currently assigned channel and on the contents of the output buffer.

If n is specified, the program examines n bytes of the output buffer in the construction of the IOCL queue. If n is not specified, the program examines parameter W bytes or, if parameter W equals zero, it only examines one byte.

The program constructs the IOCL queue using data and command chained IOCDs so that the data read is in the same relative position in the input buffer as the corresponding position of the data in the output buffer.

RB

Sets the read IOCD order byte to 06_{16} (Receive Data Blind) if Echo mode is disabled, or to $0E_{16}$ (Receive Blind with Echo) if Echo mode is enabled.

RBI

Expects a Break Interrupt on the currently assigned channel then copies and validates the device status. Break Detect is the only Test Device Status bit that the program expects if the break is an Attention function. If a transmission is in progress in Full-Duplex mode, the program expects the Break Detected, Channel End, and Unusual End status bits.

RCI

Restores the contents of the default CIB to the CIB for the currently assigned ALIM channel. If Full-Duplex is enabled in a wraparound mode, the program restores the CIBs for both channels of the Full-Duplex pair.

RD

Sets the read IOCD order byte to 02_{16} (Receive) if Echo mode is disabled, or to $0A_{16}$ (Receive with Echo) if Echo mode is enabled.

REM

Disables Echo Mode by resetting the Echo mode bit of the Read IOCD order.

RFE

Requests a framing error by complementing the character length field of the CIB of the currently assigned channel and issuing a Start I/O instruction to read four bytes. The program expects the Framing Error, Transmission Error, and Unusual End status bits.

RF='ff'

Uses the hexadecimal value ff to define a Read IOCD flag byte.

RNG

Issues a Start I/O instruction Ring Out to the currently assigned channel.

RO='ee'

Uses the hexadecimal value ee as the read IOCD order byte.

RPE

Requests a parity error by setting the Parity Enable bit, complementing the even/odd parity bit in the control information block (CIB) for the currently assigned channel and issuing a Start I/O instruction to read four bytes. The program expects the Parity Error, Transmission Error, and Unusual End status bits.

RRI

Expects a ring interrupt on the currently assigned channel.

6.6.12 "S" Directives

SBRn

Sets the Baud Rate byte in the CIB for the currently assigned channel as follows:

n	Baud Rate	Hexadecimal Code
50	50	01
75	75	02
110	110	03
134	134.5	04
150	150	05
300	300	06
600	600	07
900	900	08
1200	1200	09
1800	1800	0A
2400	2400	0B
3600	3600	0C
4800	4800	0D
7200	7200	0E
9600	9600	0F

SBRFn

Sets the Baud Rate factor bits in the control information block (CIB) of the currently assigned channel as follows:

n	Baud Rate Factor
0	Sync Mode
1	(1X)
16	(16X)
64	(64X)

SBRXn

Sets the Baud Rate byte in the CIB for the currently assigned ALIM channel to 00_{16} (external). The value of n specifies the baud rate of the external source.

SCA='nn'

Sets the Strip Character One byte in the CIB of the currently assigned ALIM channel to the hexadecimal value specified by nn.

SCB='nn'

Sets the Strip Character Two byte in the CIB of the currently assigned ALIM channel to the hexadecimal value specified by nn.

SCLn

Sets the Character Length bits in the CIB of the currently assigned channel as follows:

n	Character Length
5	5 Bits
6	6 Bits
7	7 Bits
8	8 Bits

SEM

Sets the Echo mode bit in the Read IOCD order byte.

SEP

Selects even parity by setting the Even Parity bit and the Parity Enable bit in the CIB of the currently assigned ALIM channel.

SNP

Selects no parity by resetting the Parity Enable bit in the control information block (CIB) of the currently assigned ALIM channel.

SOP

Selects odd parity by resetting the Even Parity bit and setting the Parity Enable bit in the CIB of the currently assigned channel.

SPL

Selects a private line by resetting the Private/Switched Line bit in the CIB of the currently assigned ALIM channel.

SSBn

Sets the Stop Bits field of the CIB of the currently assigned ALIM channel as follows:

n	Stop Bits
1	1 Stop Bit
15	$1\frac{1}{2}$ Stop Bits
2	2 Stop Bits

SSL

Selects a switched line by setting the Private/Switched Line bit in the CIB of the currently assigned ALIM channel.

ST

Uses the contents of parameter S to append a subtest designation to the current test number. The valid range for parameter S is one through 26 and the subtest designation appended is the letter corresponding to the number in parameter S. For example, if parameter S is equal to 1, the subtest designation is A.

STX='nn'

Sets the Start-of-Text (STX) byte in the CIB of the currently assigned ALIM channel to the hexadecimal value specified by nn.

6.6.13 "T" Directives

TBA

Issues a Test Buffer Address instruction and writes the returned status to the listed output (LO) file in the following format:

TBA STATUS (aa-c-00):00bbbbb

aa — specifies the two-character hexadecimal GPMC Controller address.

c — specifies the one-character hexadecimal ALIM channel address.

bbbbbb — specifies the IOCD buffer address maintained in the GPMC's auxiliary memory as a result of the last executed Start I/O instruction.

TCL

Performs a Test Control List Address function and writes the returned status to the LO file.

TDV

Issues a Test Device Status instruction and writes the returned status to the LO file in the following format:

TDV STATUS (aa-c-00):ddddrrrr

aa — specifies the two-character hexadecimal GPMC Controller address.

c — specifies the one-character hexadecimal ALIM channel address.

dddd — specifies the hexadecimal value of the device dependent status as maintained in the GPMC's auxiliary memory as a result of the last executed Start I/O instruction. TDV device dependent status definitions (expressed in positional notation) are:

dddd=8...$_{16}$	LINK DISCONNECTED/DATA SET NOT READY
4...$_{16}$	CARRIER OFF
2...$_{16}$	FRAMING ERROR
1...$_{16}$	RECEIVER OVERRUN
.8..$_{16}$	PARITY ERROR
.4..$_{16}$	RING DETECTED
.2..$_{16}$	BREAK DETECTED
.1..$_{16}$	NOT INITIALIZED
..8.$_{16}$	NOT USED (ALWAYS 0)
..4.$_{16}$	NOT USED (ALWAYS 0)
..2.$_{16}$	TRANSMISSION ERROR
..1.$_{16}$	INCORRECT LENGTH
...8$_{16}$	UNUSUAL END
...4$_{16}$	ILLEGAL ORDER
...2$_{16}$	INTERRUPT PENDING
...1$_{16}$	CHANNEL END

rrrr — specifies the hexadecimal value of the remaining byte count as maintained in the GPMC's auxiliary memory as a result of the last executed Start I/O instruction.

TIO

Issues a Test I/O instruction and writes the returned status (condition code transfer only) to the LO file in the following format:

TIO STATUS (aa-c-00):s

aa — specifies the two-character hexadecimal GPMC Controller address.

c — specifies the one-character hexadecimal ALIM channel address.

s — specifies the hexadecimal value of the condition code status returned. TIO condition code definitions are as follows:

$s = 8_{16}$	UNDEFINED (ALWAYS 0)
4_{16}	GPDC CHANNEL ABSENT
2_{16}	GPDC CHANNEL BUSY
1_{16}	TBA, TCLA, TDV STATUS TRANSFER NOT COMPLETE

6.6.14 "W" Directives

Wn

Issues a Start I/O Write command of n bytes or, if n is not specified, the program writes parameter W bytes.

WCI

Issues a Start I/O instruction to write control information to the currently assigned channel. In Full-Duplex mode, the program writes control information to both channels of the full-duplex pair.

The program writes control information prior to any Start I/O instruction if the control information block for the channel is changed since issuing the last Start I/O instruction.

WF='FF'

Defines the Write Interrupt flags.

WIO

Waits for I/O to complete and checks status.

WO='OO'

Defines the Write order.

6.7 Control Information

The Gould GPDC Asynchronous Line Interface Module (ALIM) Diagnostic contains three control information blocks (CIB): a transmit channel CIB designated TCIB, a receive channel CIB designated RCIB, and a default CIB designated DCIB. The transmit channel CIB contains control information used to condition the transmit channel. Similarly, the receive channel CIB contains control information used to condition the receive channel. The default CIB contains control information used by the Restore Control Information (RCI) directive to reset the transmit channel and receive channel CIBs to default (and compatible) values.

The ALIM Diagnostic's default CIB values are:

CIB Byte	Value	Description
MODE (Byte 0)	34_{16}	VRC Parity - Enable (ON) VRC Parity - Even Character Length - 8 bits Strip Sync - Enabled (Strip initial sync character on receive)
NYSN (Byte 1)	01_{16}	Number of SYNC characters - 1
FRMT (Byte 2)		Ring - Disabled (OFF) Send and Test One-Second SYNC - Disabled (OFF) Half/Full-Duplex - Half Duplex Line Control - Private Block Check Character - No BCC
DLE (Byte 3)	10_{16}	DLE character (CTRL-P)
PAD (Byte 4)	FF_{16}	PAD character
SYNC (Byte 5)	16_{16}	Sync character (CTRL-V)

The program writes control information explicitly to all currently assigned GPDC ALIM channels if executing a Write Control Information (WCI) directive. It also writes control information implicitly to a GPDC ALIM channel prior to issuing any Command Device (CD) Start I/O instructions, if the following conditions exist:

- Control information is not written previously to the controller.
- The contents of the CIB for the ALIM channel changes since last written.
- The Diagnostic Executive Program (DEXP) is reinitialized by entering RST and RUN on the operator's console.

6.8 Control Information Block (CIB) Directives

Use the following directives to modify the contents of the transmit, receive, and default CIBs:

TCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the channels transmit CIB.

RCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the channels receive CIB.

CCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of both the transmit and receive channels CIBs.

DCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the default, transmit, and receive channels CIBs.

6.9 Control Information Block (CIB) Directive Arguments

Use the following arguments with transmit, receive, and default CIBs:

NOBCC No Block Check Character

The block check character field of the specified CIB is set to the value 07_{16}.

CRC16 Select CRC-16 Block Check Character

The block check character field of the specified CIB is set to the value 00_{16}.

LRC8 Select LRC-8 Block Check Character

The block check character field of the specified CIB is set to the value 05_{16}.

CRCCCIT Select CRC-CCITT Block Check Character

The block check character field of the specified CIB is set to the value 06_{16}.

CLP(cl,p)

Sets the character length/parity (CLP) field of the specified CIB to select the character length specified by the value of "cl" and the parity specified by the optional parameter "p", as follows:

cl	p	CLP Field
6	No Parity	00_{16}
6	Even	00_{16}
6	Odd	01_{16}
7	No Parity	02_{16}
8	No Parity	03_{16}
7	Even	04_{16}
7	Odd	05_{16}
8	Even	06_{16}
8	Odd	07_{16}

DLE=xx

Sets the DLE character in the specified CIB to the hexadecimal value xx.

PAD=xx

Sets the PAD character in the specified CIB to the hexadecimal value xx.

SYN(s-0,s-1,s-2)

Sets the SYN0 character in the specified CIB to the hexadecimal value s-0, the SYN1 character to the value s-1 if specified, the SYN2 character to the value s-2 if specified, and sets the NSYN byte to the number of sync characters s-i specified.

RN/ON

Sets the ring detection enable (RN) bit of the specified CIB to a 1 (Ring Detection Enabled).

RN/OFF

Sets the ring detection enable (RN) bit of the specified CIB to a 0 (Ring Detection Disabled).

RTS/ON

Sets the inhibit RTS delay bit of the specified CIB to a 1 (RTS Delay Inhibited).

RTS/OFF

Sets the inhibit RTS delay bit of the specified CIB to a 0 (RTS Delay Enabled).

S/ON

Sets the switched line (S) bit of the selected CIB to a 1 (Switched Line).

S/OFF

Sets the switched line (S) bit of the selected CIB to a 0 (Private Line).

SN/ON

Sets the send and test one second sync (SN) bit of the specified CIB to a 1 (On).

SN/OFF

Sets the send and test one second sync (SN) bit of the specified CIB to a 0 (Off).

SS/ON

Sets the strip initial sync character (SS) bit of the specified CIB to a 1 (On).

SS/OFF

Sets the strip initial sync character (SS) bit of the specified CIB to a 0 (Off).

RC(rc_1/rc_2/../rc_n)

Appends the recognition characters rc_1 through rc_n to the specified CIB. Each recognition character field is of the form:

char:cff_1;...;cff_n

char — specifies the character to be recognized. The character is any valid ASCII character or control character (except a single quotation mark preceded by a double quotation mark). Specify ASCII control characters by preceding the character with a double quotation mark which causes a logical AND operation to be performed with the value $3F_{16}$.

cff — specifies the condition/function triplet field, and is in the form:

cond,$func_0$,$func_1$

cond — specifies a two-digit hexadecimal number representing the conditions under which the character (char) is to be recognized.

$func_0$ — specifies a two-digit hexadecimal number representing the functions in function byte 0 of the CIB.

$func_1$ — specifies a two-digit hexadecimal number representing the functions in function byte 1 of the CIB.

CHAPTER 7

I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview

The Gould GPDC Asynchronous Line Interface Module Diagnostic contains optional capabilities that provide comprehensive listings of I/O activity during program execution.

These capabilities include I/O Command Trace and Data Recorder.

By setting the appropriate options, these routines produce a visual representation to the listed output (LO) file of the data being manipulated and I/O commands being issued.

Use these listings to examine in greater detail, the processor activity by supplementing error messages and verifying that the actual data being generated coincides with the commands being issued.

7.2 I/O Command Trace

The I/O Command Trace provides a record of all I/O instructions issued and in the order the program issues them during diagnostic execution. Option 17 enables the I/O Command Trace and sends the information to the LO file.

7.2.1 Starting Execution

To begin I/O Command Trace execution, set Option 17 (enables Command Trace) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the LO file:

```
<<<<'taskname'----hh:mm:ss>>>>

TST:ttt,DEV:aa-c-00,PC:ppppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal GPMC Controller address.
c	specifies the one-character hexadecimal ALIM channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.2.2 Milestone Messages

The program writes the following milestone messages to the listed output (LO) file during an I/O Command Trace:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx — specifies the Command Device (CD) instruction issued.

qqqqqq — specifies the address of the IOCL queue if the CD instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the value.

The IOCL queue contents (repeated for all IOCDs within the IOCL queue):

```
IOCD(ii):ooffcccc 00bbbbbb
```

ii — specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo — specifies the contents of the order byte in IOCD Word 1.

ff — specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbb — specifies the hexadecimal initial buffer address in IOCD Word 2.

7.2.3 Terminating the I/O Command Trace

To terminate the I/O Command Trace, issue an Attention command and reset Option 17 (enables Command Trace) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 17
```

7.3 Data Recorder

The Data Recorder provides a dump of the actual data being manipulated during program execution. The dump shows how the data appears during I/O activity. Option 18 enables Data Recorder. With this option set, the routine sends a dump of data to the listed output (LO) file.

7.3.1 Starting Execution

To begin Data Recorder execution, set Option 18 (enables Data Recorder) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the listed output (LO) file:

```
<<<<'taskname'----hh:mm:ss>>>>

TST:ttt, DEV:aa-c-00, PC:ppppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal GPMC Controller address.
c	specifies the one-character hexadecimal ALIM channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.3.2 Milestone Messages

The program writes the following milestone messages during the Data Recorder routine:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx	specifies the Command Device (CD) instruction issued.
qqqqqq	specifies the address of the IOCL queue if the Command Device (CD) instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the address.

The I/O command doubleword (IOCD) specification for each IOCD in the I/O Command List (IOCL) queue):

```
IOCD    (ADDR)    (WRD1)      (WRD2)      ORDER IS:    order
(ii)    qqqqqq    ooffcccc    00bbbbbb    FLAGS ARE:   flag
                                                          .
                                                          .
                                                          .
                                                         flag
```

ii — specifies the queue slot number of the IOCD in the IOCL queue.

qqqqqq — specifies the address of the IOCD in the IOCL queue.

oo — specifies the contents of the order byte in IOCD Word 1.

ff — specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb — specifies the hexadecimal initial buffer address in IOCD Word 2.

order — specifies the IOCD order. If the listed output (LO) file assignment is to a console device, the program writes the hexadecimal value of the order byte. Otherwise, it writes a text description of the order:

oo=00	STOP
01	TRANSMIT DATA
02	RECEIVE DATA
03	CONNECT COMMUNICATIONS CHANNEL
05	WRITE CONTROL INFORMATION
0A	RECEIVE WITH ECHO
0B	TRANSMIT BREAK
0E	RECEIVE BLIND WITH ECHO
06	RECEIVE DATA BLIND
07	DISCONNECT COMMUNICATIONS CHANNEL
0F	RING(OUT)
80	TRANSFER-IN-CHANNEL
other	ILLEGAL

flag — specifies the IOCD flags. If the LO file assignment is to a console device, the program writes the hexadecimal value of the flag byte. Otherwise, it writes a text description of each set flag bit:

ff = 40	INTERRUPT ON ZERO BYTE COUNT
20	HALT ON TRANSMISSION ERROR
10	SUPPRESS INCORRECT LENGTH
08	INTERRUPT ON UNUSUAL END
04	COMMAND CHAIN
02	DATA CHAIN
01	INTERRUPT ON CHANNEL END

The message for each I/O command doubleword (IOCD) involving a data transfer:

```
IOCD  (ADDR)    DATA LENGTH SPECIFIED: ccccc TRANSFERRED: tttt
(ii)  bbbbbb    mmm mmm a a a . . a a mmm mmm
      (Rnnnnn)  xx..xx..xxxxxx....xxxxxx..xx.
```

ccccc — specifies the decimal value of initial byte count in IOCD Word 1.

ttttt — specifies the number of decimal bytes actual transferred (only written for the last executed IOCD in the I/O Command List (IOCL) queue).

ii — specifies the queue slot number of the IOCD in the IOCL queue.

bbbbbb — specifies the hexadecimal initial buffer address in IOCD Word 2.

Rnnnnn — specifies the relative frame number of the first data byte in the line.

mmm — specifies the mnemonic name of an ASCII control character associated with the hexadecimal representation of the ASCII data byte ..xx.. :

mmm=STX	Start-of-Text
ETX	End-of-Text
CHA	Termination Character A
CHB	Termination Character B
CHC	Termination Character C
SC1	Strip Character 1
SC2	Strip Character 2

a — specifies the graphic character associated with the hexadecimal representation of the ASCII data byte ..xx.. .

..xx.. — specifies the hexadecimal representation of an ASCII data byte.

The message for the terminal IOCD:

```
IOCD   ADDR:   DEVICE STATUS AT TERMINAL IOCL: status
(ii)   qqqqqq                                    .
                                                 .
                                                 .
                                               status
```

ii — specifies the queue slot number of the last executed IOCD in the IOCL queue.

qqqqqq — specifies the address of the last executed IOCD in the IOCL queue.

status — specifies the test device (TDV) status at I/O completion. If the value of the TDV status is 0000_{16}, the program writes one of the following messages:

I/O COMPLETE	Normal completion.
I/O TIME-OUT	I/O not completed within expected time, HALT I/O is issued to terminate operation.

If the value of the test device (TDV) status is other than 0000_{16} and the listed output (LO) file assignment is to a console device, the program writes the hexadecimal value of the TDV status. Otherwise, it writes a text description of each TDV set status bit:

```
8...  LINK DISCONNECTED
4...  CARRIER OFF
2...  FRAMING ERROR
1...  RECEIVER OVERRUN
.8..  PARITY ERROR
.4..  RING DETECTED
.2..  BREAK DETECTED
.1..  CHANNEL INOP/NOT INITIALIZED
..8.  N/A
..4.  N/A
..2.  TRANSMISSION ERROR
..1.  INCORRECT LENGTH
...8  UNUSUAL END
...4  ILLEGAL ORDER
...2  INTERRUPT PENDING
...1  CHANNEL END
```

7.3.3 Terminating Data Recorder

To terminate the Data Recorder routine, issue an Attention command and reset Option 18 (enables Data Recorder) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 18
```

Gould GPDC Binary Synchronous Line Interface Module

(GP.BLM.r)

Diagnostic Description

Diagnostic Number: 001032-984r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005750-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould GPDC Binary Synchronous Line Interface Module (GP.BLM.r) Diagnostic Description Release 1.0, Publication Order Number **326-005750-000,** was printed May 1985.

This manual contains the following pages:

Title page
Copyright page
iii/iv through vii/viii
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-2
4-1 through 4-4
5-1 through 5-7/5-8
6-1 through 6-19/6-20
7-1 through 7-6

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Model Number Assignments 2-2
2.3 Program Option Set 2-2

CHAPTER 3 PROGRAM EXECUTION

3.1 Program Execution 3-1
3.2 Milestone Messages 3-1
3.3 Terminating Program Execution 3-2

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Message Routines 4-1
4.2 Unique Error Messages 4-3

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Link Disconnected/Data Set Not Ready 5-1
5.3 Test 2 Channel Inoperable Status 5-2
5.4 Test 3 Illegal Order Status 5-2
5.5 Test 4 Header Text Protocol 5-2
5.6 Test 5 Message Text Protocol 5-2
5.7 Test 6 Combined Header/Message Text Protocol 5-2
5.8 Test 7 Transparent Text Protocol 5-3
5.9 Test 8 Transparent Transmission - All Zeros 5-3
5.10 Test 9 Transparent Transmission - All Ones 5-3
5.11 Test 10 Transparent Transmission - Alternating As and 5s 5-3
5.12 Test 11 Transparent Transmission - Binary Progression 5-3
5.13 Test 12 Transparent Transmission - ASCII Subtest 5-3
5.14 Test 13 Transparent Transmission - Floating One 5-3
5.15 Test 14 Transparent Transmission - Floating Zero 5-4
5.16 Test 15 Transparent Transmission - Random Data 5-4
5.17 Test 16 Combined Header/Transparent Text Protocol 5-4
5.18 Test 17 Intermediate Text Block Protocol 5-4
5.19 Test 18 Message Response Protocol 5-4
5.20 Test 19 Imbedded Termination Control Character Sequences 5-4
5.21 Test 20 Data Link Escape and Synchronous Idle Transmission Protocol 5-5
5.22 Test 21 Polling Sequence Protocol 5-5
5.23 Test 22 Variable Transfer Length 5-5
5.24 Test 23 Programmable Vertical Parity Selection 5-5
5.25 Test 24 Vertical Parity Error Detection 5-5
5.26 Test 25 Programmable Character Length Selection 5-5
5.27 Test 26 Programmable Baud Rate Selection 5-6
5.28 Test 27 Ring Interrupt Detection 5-6
5.29 Test 28 Programmable Code Conversion 5-6
5.30 Test 29 Programmable Block Check Character Selection 5-7
5.31 Test 30 Block Check Character Error Detection 5-7
5.32 Test 31 Random I/O Command List Sequence 5-7

CHAPTER 6 DIRECTIVE LANGUAGE

6.1 Directive Overview 6-1
6.2 Using Directives 6-1
6.3 Using Parameters 6-2
6.4 Using Arguments 6-4
6.5 DEXP Directive Summary 6-5
6.6 Gould GPDC Binary Synchronous Line Interface Module Diagnostic Unique Directives 6-8
6.6.1 "A" Directives 6-8
6.6.2 "B" Directives 6-8
6.6.3 "C" Directives 6-9
6.6.4 "D" Directives 6-9
6.6.5 "H" Directive 6-10
6.6.6 "I" Directive 6-10
6.6.7 "M" Directive 6-11
6.6.8 "R" Directives 6-11
6.6.9 "S" Directive 6-11
6.6.10 "T" Directives 6-12
6.6.11 "W" Directive 6-14
6.6.12 "X" Directives 6-14
6.7 Control Information 6-15
6.8 Control Information Block (CIB) Directives 6-17
6.9 Control Information Block Directive Arguments 6-17

CHAPTER 7 I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview 7-1
7.2 I/O Command Trace 7-1
7.2.1 Starting Execution 7-1
7.2.2 Milestone Messages 7-2
7.2.3 Terminating I/O Command Trace 7-2
7.3 Data Recorder 7-3
7.3.1 Starting Execution 7-3
7.3.2 Milestone Messages 7-3
7.3.3 Terminating Data Recorder 7-6

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould GPDC Binary Synchronous Line Interface Module (GP.BLM.r) Diagnostic is a level two macrodiagnostic program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 31 tests, which verify the correct operation of GPDC instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring operator intervention by setting the appropriate options.

During execution, the program writes milestone messages and error message sequences to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode). The CC file assignments are to disc or tape and the OC file assignments are to the operator's console or a terminal configured on an eight-line asynchronous controller.

1.2 Resource Requirements

The Gould GPDC Binary Synchronous Line Interface Module (BLIM) Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory
- A GPDC BLIM Controller, Model 9116
- A GPMC IOM Controller, Model 9103, 16 megabyte, or
  A GPMC IOM Controller, Model 9104, 512 kilobyte

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould GPDC Binary Synchronous Line Interface Module (GP.BLM.r) Diagnostic provides the capability of testing from two to 16 GPDC BLIM channels connected in a Half-Duplex configuration. The minimum configuration consists of a pair of GPDC BLIM channels connected to a single General Purpose Multiplex Controller (GPMC).

The Gould BLIM Diagnostic is a self-initializing program and sets up its default values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the appropriate Diagnostic Facility User's Manual. The program name for loading the Gould BLIM Diagnostic is GP.BLM.r.

The Gould BLIM Diagnostic sets up the following default assignments:

- A jumpered Half-Duplex configuration
- A GPMC Controller address of 20
- A GPDC BLIM channel address of 02 jumpered to 03

For systems configured Full-Duplex, sufficient hardware exists to verify both GPDC BLIM channels (with or without a modem). However, for systems configured Half-Duplex, a second GPDC BLIM channel is required for program testing purposes.

For systems configured with a modem, execute the program in a jumpered Half-Duplex (JH) configuration first and then in a modem Half-Duplex (MH) configuration.

To modify the default configurations, the GPMC IOM Controller address, and/or the GPDC BLIM channel addresses, use the DEXP Assign command with the following device type codes:

Code	Description
JH	Jumpered half-duplex
MH	Modem half-duplex

For Half-Duplex configurations, the primary transmit channel, which is designated P(TX), is jumpered to the primary receive channel, which is designated P(RX). The BLIM Diagnostic assigns the odd numbered channels as primary receive channels and the even numbered channels as primary transmit channels. The program maintains this relationship for one program pass, after which the program swaps the relationship. If Option 19 (inhibits channel swapping) is set, the program does not swap channels.

2.2 Model Number Assignments

The Gould Binary Synchronous Line Interface Module (BLIM) Diagnostic sets up the default model number assignment as 9103 (a GPMC IOM Controller capable of addressing 16 megabytes of memory). To change the model number assignment to 9104 (a GPMC IOM Controller capable of addressing 512K bytes of memory), use the MOD directive.

After assigning the model number, the program writes the following message to the operator communications (OC) file:

```
'taskname' WHAT NEXT??
```

2.3 Program Option Set

Once loaded, the Gould GPDC Binary Synchronous Line Interface Module Diagnostic enters a wait state.

Using the DEXP, Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if an error is detected. Prior to suspension, the program writes an associated error message and issues the following prompt: `'taskname' ERROR... ;[C,H,T]?` Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option repeats the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their function occur prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.

Option	Default	Effect If Set
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Inhibits operator intervention tests. Setting this option inhibits any test requiring operator intervention.
9	Reset	Enables Data Recorder on error.
10	Set	Writes pass count messages. Setting this option writes the program's pass count messages to the listed output (LO) file.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone messages to the listed output (LO) file.
12	Reset	Inhibits error message expansion.
13	Reset	Inhibits status error messages.
14	Reset	Inhibits test buffer address (TBA) error messages.
15	Reset	Inhibits data error messages.
16	Reset	Inhibits test command list address (TCLA) error messages.
17	Reset	Enables Command Trace.
18	Reset	Enables Data Recorder.
19	Reset	Inhibits the P(TX)/P(RX) channel swap during program execution.

CHAPTER 3

PROGRAM EXECUTION

3.1 Program Execution

After setting options, issue a DEXP Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-984r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Milestone Messages

The program verifies the device assignments made. If the device type code is other than JH or MH, the program writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED: INVALID DEVICE TYPE CODE tt
```

Reassign the device to be tested using one of the valid device type codes.

The program then verifies that a pair of GPDC BLIM channels are specified. If only a single channel is specified, the program writes the following message to the OC file:

```
'taskname' TERMINATED: DEVICE ASSIGNMENT ERROR
```

Reassign the correct pairs of GPDC BLIM channels.

The program then verifies the presence of the GPMC IOM Controller and, if it is absent, the program writes the following message to the OC file:

```
'taskname' TERMINATED: GPMC:aa ABSENT
```

Reassign the device to be tested, using the correct GPMC IOM Controller address.

The program then verifies the presence of the GPDC BLIM channels specified and, if they are absent, the program writes the following message to the OC file:

```
'taskname' TERMINATED: GPDC:c ABSENT
```

Reassign the GPDC BLIM channels to be tested using correct channel addresses.

The program then sequentially executes all designated tests for each assigned GPDC Controller.

If Option 11 (writes milestone messages) is set, the program writes the following message to the listed output (LO) file prior to executing each test:

```
hh:mm:ss 'taskname' TEST: n
```

If Option 10 (writes pass count messages) is set, the program writes the following message to the LO file after completing all designated tests:

```
hh:mm:ss 'taskname' FINISHED PASS: n
```

3.3 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system, causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Message Routines

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic verifies the correct operation of the controller (I/O command) function through a fixed sequence of checks. If the program detects an error and Option 0 (inhibits error messages) is reset, it writes a message describing actual and expected results during execution of an I/O command.

The program writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

```
<<<<'taskname'----hh:mm:ss>>>>
```

taskname	specifies the one to eight character program name indicated in the load map.
hh:mm:ss	specifies the system elapsed execution time at the point of error.

2. The error message header

```
TST:ttt, DEV:aa-c-00, PC:ppppp
```

ttt	specifies the test number of the erring test.
aa	specifies the two-character hexadecimal GPMC IOM Controller address.
c	specifies the one-character hexadecimal BLIM channel address.
ppppp	specifies the program counter value within the program.

3. The erring directive (for directive tests only)

```
DRTV: directive
```

directive	specifies the erring directive.

4. The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable)

```
CMND:xxxxxxxx, IOCLQP:00qqqqq
```

xxxxxxxx specifies the Command Device (CD) instruction issued.

qqqqq specifies the address of the IOCL queue if the Command Device (CD) instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the value.

5. The IOCL queue contents

```
IOCD(ii):ooffcccc      00bbbbbb
```

ii specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo specifies the contents of the order byte in IOCD Word 1.

ff specifies the contents of the flag byte in IOCD Word 1.

cccc specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb specifies the initial buffer address in IOCD Word 2.

4.2 Unique Error Messages

```
IOCD(ii)STATUS ERROR(S)
EXP(DV):ddddcccc     (BA):00bbbbbb (CLA):00aaaaaa
ACT(DV):ddddcccc     (BA):00bbbbbb (CLA):00aaaaaa
```

ii — specifies the queue slot number of the I/O command doubleword (IOCD) within the I/O Command List (IOCL) queue where the status error occurred.

EXP — indicates the expected status values.

ACT — indicates the actual status values.

dddd — specifies the hexadecimal value of the test device (TDV) device status.

cccc — specifies the hexadecimal value of the TDV remaining byte count.

bbbbbb — specifies the hexadecimal value of the test buffer address (TBA) status.

aaaaaa — specifies the hexadecimal value of the test command list address (TCLA) status.

If Option 12 (inhibits error message expansion) is reset, the program writes one of the following messages for each status bit using a plus sign to indicate an unexpected status bit and a minus sign to indicate a missing status bit in either the actual or expected status reports:

```
8...  LINK DISCONNECTED/DATA SET NOT READY
4...  BCC ERROR
2...  FORMAT ERROR
1...  RECEIVE OVERRUN
.8..  VRC ERROR
.4..  (R.) RING DETECTED (T.) NOT C2S
.2..  LOSS OF SYNC
.1..  CHANNEL INOPERABLE/NOT INITIALIZED
..8.  N/A
..4.  N/A
..2.  TRANSMISSION ERROR
..1.  INCORRECT LENGTH
...8  UNUSUAL END
...4  ILLEGAL ORDER
...2  INTERRUPT PENDING
...1  CHANNEL ENDING
```

If the program executes a specified instruction but receives no service interrupt, it writes the following message:

```
I/O TIMEOUT (NO SI)
```

If data comparison discrepancies occur, the program writes the following message:

```
IOCD(ii):DATA ERROR(S)
FR#: EXP: ACT:
ffff  dd   dd
  .    .    .
  .    .    .
  .    .    .
ffff  dd   dd
xxxx OUT OF yyyy
```

ii — specifies the queue slot number of the I/O command doubleword (IOCD) within the I/O Command List (IOCL) queue where the data errors occurred.

FR# — indicates the frame numbers in error.

EXP — indicates the expected input buffer value.

ACT — indicates the actual input buffer value.

ffff — specifies the frame number in error relative to the first frame of the IOCD where the error occurred (a maximum of five error frames are reported for each IOCD).

dd — specifies the hexadecimal value of the expected or actual data element.

xxxx — specifies the total number of frames in error for the IOCD.

yyyy — specifies the total number of frames compared for the IOCD.

The total number of frames compared by the program is equal to the number of bytes specified in the initial byte count for each IOCD. However, for the terminal IOCD, if the test device remaining byte count indicates that not all bytes requesting transfer are transferred, the program compares only those bytes actually transferred.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould GPDC Binary Synchronous Line Interface Module (GP.BLM.r) Diagnostic contains a sequence of 31 tests which verify unique GPDC BLIM channel functions or line discipline protocol. The program selects a subset of the 31 tests based on the device type code specified in the DEXP Assign command. The assignment mode for each test is noted under the individual test heading.

The subset of tests selected executes in a continuous sequence or individually by operator request.

Skip tests by selecting appropriate options.

5.2 Test 1 Link Disconnected/Data Set Not Ready

Execution Mode: JH

This test contains seven subtests which verify that a BLIM channel detects Link Disconnected and Data Set Not Ready conditions.

The program writes the following message to the operator communications (OC) file:

```
'taskname' SUSPENDED:  DISCONNECT JUMPER CABLE DEV: aa-c
```

Disconnect the jumper cable connected to GPDC BLIM channel "c" on GPMC Controller "aa" and continue program execution by issuing a DEXP Continue command.

After subtest 1G is completed, the program writes the following message to the OC file:

```
'taskname' RECONNECT JUMPER CABLE DEV: aa-c
```

The program then repeats Subtests 1A through 1G on the second GPDC BLIM channel in the currently assigned BLIM pair.

The program skips this test if Option 7 (runs directive tests only) or Option 8 (inhibits operator intervention tests) are set.

5.3 Test 2 Channel Inoperable Status

Execution Mode: JH

This test contains eight subtests which verify that a BLIM channel detects a Channel Inoperable/Not Initialized condition.

Execution of Subtest 2B forces the errors reported in subtests 2C through 2F and simulates a condition similar to an unissued Write Control Information command.

A Write Control Information order is invalid if the transfer length is not exactly six bytes. CIBs written by the program are 34 bytes in length, which is sufficient to define all commonly used control character sequences.

Subtests 2A through 2H are repeated for each GPDC BLIM channel in the currently assigned BLIM pair.

The program skips this test if Option 7 (runs directive tests only) is set.

5.4 Test 3 Illegal Order Status

Execution Mode: JH

This test verifies that the GPDC controller detects an illegal order (undefined function) and posts the correct test device (TDV) status.

The program skips this test if Option 7 (runs directive tests only) is set.

5.5 Test 4 Header Text Protocol

Execution Modes: JH, MH

This test contains three subtests which verify the protocol associated with the transmission of a message header (text block started with a Start-of-Heading (SOH) control character).

Then, the subtest constructs the I/O Command List (IOCL) queues and initiates the I/O transfer.

5.6 Test 5 Message Text Protocol

Execution Modes: JH, MH

This test contains three subtests which verify the protocol associated with the transmission of a message (text block started with a Start-of-Text (STX) control character).

5.7 Test 6 Combined Header/Message Text Protocol

Execution Modes: JH, MH

This test contains three subtests which verify the protocol associated with the transmission of a message header (text block started with the Start-of-Heading (SOH) control character), followed by the message (text block started with a Start-of-Text (STX) control character).

5.8 Test 7 Transparent Text Protocol

Execution Modes: JH, MH

This test contains three subtests which verify the protocol associated with the transmission of a transparent message (text block started with a Data Link Escape/ Start-of-Text (DLE-STX) control character sequence).

5.9 Test 8 Transparent Transmission - All Zeros

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block of zeros to detect picked bits.

5.10 Test 9 Transparent Transmission - All Ones

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block of ones to detect dropped bits.

5.11 Test 10 Transparent Transmission - Alternating As and 5s

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block consisting of an alternating bit pattern.

5.12 Test 11 Transparent Transmission - Binary Progression

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block consisting of a binary progression to ensure taht the controller transmits each possible eight-bit binary byte.

5.13 Test 12 Transparent Transmission - ASCII Subset

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block consisting of the 64-character printable ASCII subset.

5.14 Test 13 Transparent Transmission - Floating One

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block consisting of a one bit floating through a field of zeros to ensure that the controller transmits a one bit in each bit position of a byte.

5.15 Test 14 Transparent Transmission - Floating Zero

Execution Modes: JH, MH

This test verifies that the BLIM transmits and receives a data block consisting of a zero bit floating through a field of ones to ensure that the controller transmits a zero bit in each bit position of a byte.

5.16 Test 15 Transparent Transmission - Random Data

Execution Mode: JH

This test verifies that the BLIM transmits and receives a data block consisting of random data to detect pattern sensitivity.

5.17 Test 16 Combined Header/Transparent Text Protocol

This test contains three subtests which verify the protocol associated with the transmission of a message header (text block started with a Start-of-Header (SOH) control character) followed by a transparent message (text block started with a Data Link Escape/Start-of-Text (DLE-STX) control character sequence).

5.18 Test 17 Intermediate Text Block Protocol

Execution Mode: JH

This test contains three subtests which verify the protocol associated with the transmission of messages containing imbedded Intermediate Text Block (ITB) control characters. The ITB control character forces transmission of the accumulated block check characters (BCC) and requires that the transmit channel perform a Command Chain Transmit Data-to-Transmit Data order and the receive channel performs a Command Chain Receive Data-to-Receive Data order.

5.19 Test 18 Message Response Protocol

Execution Mode: JH

This test contains five subtests which verify the protocol associated with the transmission of the message response control characters Positive Acknowledge (ACK0) and (ACK1), Wait Before Transmit Positive Acknowledge (WACK), Reverse Channel Interrupt (RVI), and Negative Acknowledge (NAK).

5.20 Test 19 Imbedded Termination Control Character Sequences

Execution Mode: JH

This test contains four subtests which verify the protocol associated with the transmission of the message termination control characters Enquiry (ENQ), End-of-Transmission (EOT), End-of-Text-Block (ETB), and End-of-Text (ETX) if they are imbedded within a transparent data block.

5.21 Test 20 Data Link Escape and Synchronous Idle Transmission Protocol

Execution Mode: JH

This test contains two subtests which verify the protocol associated with the continuous transmission of the control characters Data Link Escape (DLE) and Synchronous Idle (SYN) if they are imbedded within a transparent data block.

5.22 Test 21 Polling Sequence Protocol

Execution Mode: JH

This test verifies the protocol associated with the transmission of the basic polling sequence.

5.23 Test 22 Variable Transfer Length

Execution Mode: JH

This test verifies that the BLIM transmits and receives records of varying length.

5.24 Test 23 Programmable Vertical Parity Selection

Execution Mode: JH

This test contains three subtests which verify the programmable vertical parity selection functions.

5.25 Test 24 Vertical Parity Error Detection

Execution Mode: JH

This test contains four subtests which verify that the BLIM receives vertical parity errors if noncompatible vertical parity modes are selected.

Note: In Subtests 24A and 24C, a vertical parity error should occur. In Subtests 24B and 24D, no vertical parity error should occur.

5.26 Test 25 Programmable Character Length Selection

Execution Mode: JH

This test contains three subtests which verify the programmable character length selection functions of the BLIM.

5.27 Test 26 Programmable Baud Rate Selection

Execution Mode: JH

This test contains 15 subtests which verify the programmable baud rate selection functions of the BLIM and the internal baud rate generator.

The program increments the selected baud rate and repeats the transfer until generating the following baud rates:

Subtest	Baud Rate	Transfer Length (bytes)
A	150 bps	256
B	300 bps	256
C	600 bps	256
D	900 bps	256
E	1200 bps	512
F	1800 bps	768
G	2400 bps	1024
H	3600 bps	1280
I	4800 bps	1576
J	7200 bps	1792
K	9600 bps	2048
L	14.4 Kbps	2048
M	19.2 Kbps	2048
N	28.8 Kbps	2048
O	38.4 Kbps	2048

5.28 Test 27 Ring Interrupt Detection

Execution Mode: JH

This test verifies the ability of BLIM to transmit and receive a ring interrupt.

5.29 Test 28 Programmable Code Conversion

Execution Mode: JH

This test contains three subtests which verify the programmable code conversion functions of the BLIM.

The code converter states selected results in the following data patterns:

Subtest	Output Buffer Data	Input Buffer Data
28A	8-bit binary	8-bit binary
28B	8-bit binary	EBCDIC
28C	8-bit binary	ASCII

5.30 Test 29 Programmable Block Check Character Selection

Execution Mode: JH

This test contains six subtests which verify the programmable block check character (BCC) selection functions of the BLIM.

5.31 Test 30 Block Check Character Error Detection

Execution Mode: JH

This test contains three subtests which verify that the BLIM receives a block check character (BCC) error if noncompatible BCC modes are selected.

5.32 Test 31 Random I/O Command List Sequence

Execution Mode: JH

This test verifies that the BLIM channels correctly respond to any combination of randomly generated I/O Command List (IOCL) sequences. The test is entirely data driven. That is, the contents of the output buffer define the IOCL sequence created. Each IOCL sequence (transmit and receive) may contain up to 32 individual IOCDs command-chained and data-chained together. The IOCL queues need not be of the same length.

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform operations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

```
'taskname' ACTIVE [C,T]?
```

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

```
DXP>taskname LA,CR,DO <cr>
```

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

```
DXP>BATCH bfile
```

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

```
DXP>taskname MAC,mfile
```

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

```
DXP>PK*L
```

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Other operand values are used such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould GPDC Binary Synchronous Line Interface Module Diagnostic parameters and a summary of their functions follow:

Parameter	Use	Description
A	Reserved	Contains the input/output buffer address and address attributes (F=1,C=00).
B	Reserved	Contains the baud rate to be used for I/O transfer time calculation (default rate is 9600 baud).
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PD directive.
E	Available	Available for general use.
F	Reserved	Contains the program control flags: Bit 29=0: Split channel configuration Bit 29=1: Wraparound channel configuration Bit 30=0: Data Terminal Equipment Bit 30=1: Data Communications Equipment (Modem) Bit 31=0: Half-Duplex Bit 31=1: Full-Duplex
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Reserved	Contains the status mask pattern: B00 through 15: TDV Device Status Mask B16 through 31: TDV Remaining Byte Count Mask
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.

Parameter	Use	Description
R	Reserved	Default record count.
S	Reserved	Contains a number between 1 and 26 corresponding to a letter from A through Z it is used by the ST directive as a subtest designator.
T	Available	Available for general use.
U	Available	Available for general use.
V	Reserved	Contains the default transfer count in bytes.
W	Reserved	Contains the transfer count to be used by the XFR directive.
X	Reserved	Contains the implied hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for JZ directives.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the program's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn	
AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L=' ' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the last random number pattern.
LNRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

Pp (op) k

Pp_1 (op) Pp_2

PD (op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PI (op) $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

PZ= $\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould GPDC Binary Synchronous Line Interface Module Diagnostic Unique Directives

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic contains unique directives which perform functions specific to the program.

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic unique directives follow.

6.6.1 "A" Directives

AL='string'

Appends the literal, indicated by 'string', to the output buffer.

APR

Inserts the BLIM channel address associated with the primary receive channel into the device definition table (DDT) and then creates the Command Device (CD) and Test Device (TD) instructions necessary to communicate with the unit under test.

APT

Inserts the BLIM channel address associated with the primary transmit channel into the DDT and then creates the CD and TD instructions necessary to communicate with the unit under test.

6.6.2 "B" Directives

BFI='address'

Relocates the program's input buffer to the specified address. The address may be any address within 16 megabytes of memory if a Model 9103 GPMC is configured, or anywhere within 512 kilobytes of memory if a Model 9104 GPMC is configured. The length of the relocated buffer equals the default program buffer length.

BFI='PGM'

Relocates the program's input buffer back to the default location within the program.

BFO='address'

Relocates the program's output buffer to the specified address. The address may be any address within 16 megabytes of memory if a Model 9103 GPMC is configured, or anywhere within 512 kilobytes of memory if a Model 9104 GPMC is configured. The length of the relocated buffer equals the default program buffer length.

BFO='PGM'

Relocates the program's output buffer back to the default location within the program.

6.6.3 "C" Directives

CCH

Issues a Start I/O instruction to transmit a Connect Communications Channel order (03_{16}) to the currently assigned BLIM channels. If required, the program performs an implicit Write Control Information function prior to issuing the Connect Communications Channel order and writes the following message to the operator communications (OC) file:

```
'taskname' CONNECT COMMUNICATIONS LINK [C,T]:
```

Connect the jumper cable (for jumpered half-duplex assignments) or the modems (for modem half-duplex assignments) before continuing execution. Respond with C to continue program execution, or with T to terminate execution.

CCIB='arg_1,...,arg_n'

Modifies the contents of the transmit and receive channel control information blocks (CIB) according to the arguments 'arg_1,...,arg_n'.

CIB

Writes the contents of the CIB for the assigned BLIM channel to the listed output (LO) file.

6.6.4 "D" Directives

DCB

Writes the contents of the device control block (DCB) for the assigned BLIM channel to the listed output (LO) file. Device control block elements are:

Word 0	FLAG:	Flag Word
Word 1	TLIS:	Tests Last Interrupt Status (this cell is used only in the DCB for GPDC 0 for each GPMC)
Word 2	CMND:	Last command issued to the GPDC
Word 3	CLQP:	IOCL Queue Pointer
Word 4	DCEP:	DCB Extension (DCE) Pointer
Word 5	TDVS:	Test Device Status
Word 6	TCLA:	Test Command List Address Status
Word 7	TBAS:	Test Buffer Address Status

DCE

Writes the contents of the device control block extension (DCE) for the assigned BLIM channel to the listed output (LO) file. Device control block extension elements are:

Word 0	FLAG:	Flag Word
Word 1	CIBP:	Control Information Block (CIB) Pointer
Word 2	TBCA:	Transfer Byte Count Accumulator (sum of individual initial byte counts)
Word 3	CLQP:	IOCL Queue Pointer
Word 4	IOTT:	I/O Termination System Elapsed Time
Word 5	TDVS:	Expected Test Device Status
Word 6	TCLA:	Expected Terminal IOCL Address
Word 7	TBAS:	Expected Terminal Buffer Address Status

DCH

Issues a Start I/O instruction to transmit a Disconnect Communications Channel order (07_{16}) to each assigned BLIM channel.

DCIB

Writes the contents of the default control information block (DCIB) to the LO file.

DCIB='arg_1,...,arg_n'

Modifies the contents of the default, transmit, and receive channel control information blocks (CIB) according to the specified arguments 'arg_1,...,arg_n'.

DQ

Writes the contents of the I/O Command List (IOCL) queue for the assigned BLIM channel to the LO file in the following format:

```
IOCD(ii):ooffcccc 00bbbbbb
```

ii	specifies the IOCD queue slot number in the IOCL queue.
oo	specifies the contents of the order byte in IOCL Word 1.
ff	specifies the contents of the flag byte in IOCD Word 1.
cccc	specifies the hexadecimal value of the initial byte count in IOCD Word 1.
bbbbbb	specifies the hexadecimal value of the initial buffer address in IOCD Word 2.

If no IOCL queue entry exists, the program writes the following message:

```
IOCL QUEUE EMPTY
```

6.6.5 "H" Directive

HIO

Issues a Halt I/O instruction to the currently assigned BLIM channel.

6.6.6 "I" Directive

IOR

Issues a I/O Reset instruction.

CAUTION

An I/O Reset instruction causes the program to generate a master clear for all GPDC controllers configured on the assigned GPMC. Therefore, executing an I/O Reset in one GPDC program can cause errors to occur in other GPDC programs executing concurrently.

6.6.7 "M" Directive

MOD
MODn

Aligns all internal parameters consistent with the model number n, if specified. If n is not specified, the program issues the default model number (9103).

n	Parameters
9103	use 16-megabyte GPMC parameters
9104	use 512-kilobyte GPMC parameters

6.6.8 "R" Directives

RCI

Restores the contents of the control information blocks (CIB) for the transmit and receive channels to their default values.

RCIB

Writes the contents of the CIB for the BLIM channel currently assigned as the receive channel to the listed output (LO) file.

RCIB='arg_1,...,arg_n'

Modifies the contents of the receive channel CIB according to the specified arguments 'arg_1,...,arg_n'.

RDCB

Writes the contents of the device control block (DCB) for the BLIM channel currently assigned as the receive channel to the LO file. (Refer to the DCB directive for display format).

RDCE

Writes the contents of the device control block extension (DCE) for the BLIM channel currently assigned as the receive channel to the LO file. (Refer to the DCE directive for display format.)

6.6.9 "S" Directive

ST

Uses the contents of parameter S to append a subtest designation to the current test number. Parameter S must lie in the range of 1 through 26, and the subtest designation is the letter corresponding to the number in parameter S (e.g., if S is equal to one, the subtest designation is A).

6.6.10 "T" Directives

TBA

Issues a Command Device (CD) Test Buffer Address and writes the returned status to the listed output (LO) file in the following format:

```
TBA STATUS (aa-c-00):00bbbbbb
```

aa — specifies the two-character hexadecimal GPMC IOM Controller address.

c — specifies the one-character hexadecimal BLIM channel address.

bbbbbb — specifies the current buffer address maintained in the GPMC's auxiliary memory as a result of the last Start I/O instruction executed.

TCIB

Writes the contents of the control information block (CIB) for the BLIM channel currently assigned as the transmit channel to the LO file.

TCIB='arg_1,...,arg_n'

Modifies the contents of the transmit channel CIB according to the specified arguments 'arg_1,...,arg_n'.

TCL

Issues a Test Control List Address instruction and writes the returned status to the LO file in the following format:

```
TCL STATUS (aa-c-00):00cccccc
```

aa — specifies the two-character hexadecimal GPMC IOM Controller address assigned.

c — specifies the one-character hexadecimal BLIM channel address.

cccccc — specifies the current IOCL address maintained in the GPMC's auxiliary memory as a result of the last Start I/O instruction executed.

TDCB

Writes the contents of the device control block (DCB) for the BLIM channel currently assigned as the transmit channel to the LO file (refer to the DCB directive for display format).

TDCE

Writes the contents of the device control block extension (DCE) for the BLIM channel currently assigned as the transmit channel to the LO file (refer to the DCE directive for display format).

TDV

Issues a Test Device (TDV) Status instruction and writes the returned status to the listed output (LO) file in the following format:

```
TDV STATUS (aa-c-00):ddddcccc
```

aa — specifies the two-character hexadecimal GPMC IOM Controller address.

c — specifies the one-character hexadecimal BLIM channel address.

dddd — specifies the hexadecimal value of the current device-dependent status maintained in the GPMC's auxiliary memory as a result of the last Start I/O instruction executed. TDV device-dependent status definitions (expressed in positional notation) are as follows:

```
dddd = 8...  LINK DISCONNECTED/DATA SET NOT READY
       4...  BCC ERROR
       2...  FORMAT ERROR
       1...  RECEIVER OVERRUN
       .8..  VRC ERROR
       .4..  (RX) RING DETECTED/(TX) NOT C2S
       .2..  LOSS OF SYNC
       .1..  CHANNEL INOPERABLE/NOT INITIALIZED
       ..8.  N/A
       ..4.  N/A
       ..2.  TRANSMISSION ERROR
       ..1.  INCORRECT LENGTH
       ...8  UNUSUAL END
       ...4  ILLEGAL ORDER
       ...2  INTERRUPT PENDING
       ...1  CHANNEL END
```

cccc — specifies the hexadecimal value of the current remaining byte count maintained in the GPMC's auxiliary memory as a result of the last Start I/O instruction executed.

TIO

Issues a Test I/O instruction and writes the returned status (condition code transfer only) to the LO file in the following format:

```
TIO STATUS (aa-b-00):c
```

aa — specifies the two-character hexadecimal GPMC IOM Controller address.

b — specifies the one-character hexadecimal BLIM channel address.

c — specifies the hexadecimal value of the condition code status returned:

```
c = 8  UNDEFINED (ALWAYS 0)
    4  GPDC CHANNEL ABSENT
    2  GPDC CHANNEL BUSY
    1  TBA, TCLA, TDV STATUS TRANSFER NOT COMPLETE
```

6.6.11 "W" Directive

WCI
WCIn

Writes n or CIB.LGTH bytes of control information to the assigned BLIM channels. If the BLIM channels assignments are modem half-duplex (data communications equipment mode, switched line) and the communications link is connected (refer to the CCH directive), the program writes the following message to the operator communications (OC) file:

```
'taskname' WCI WILL DISCONNECT LINK [C,T]:
```

Respond with C to continue execution, or with T to terminate execution. Reconnect the communication link following execution of the WCI directive.

6.6.12 "X" Directives

XFR
XFRn

Constructs the I/O Command List (IOCL) queues for the transmit and receive channels and subsequently issues Start I/O instructions to initiate data transfer.

If n is specified, the program examines n bytes of the output buffer. If n is not specified, the program examines parameter W bytes or, if parameter W equals zero, it examines only one byte.

XFR='BCCERROR'

Performs all of the XFR functions but expects terminal receive channel status of Unusual End and BCC Error.

XFR='VRCERROR'

Performs all of the XFR functions but expects terminal receive channel status of Unusual End and VRC Error.

XFR='RING'

Issues a Start I/O instruction to transmit a ring.

6.7 Control Information

The Gould Binary Synchronous Line Interface Module (BLIM) Diagnostic contains three control information blocks (CIB): a transmit channel CIB (TCIB), a receive channel CIB (RCIB), and a default CIB (DCIB).

The transmit channel CIB contains control information that conditions the transmit channel.

The receive channel CIB contains control information that conditions the receive channel.

The default CIB contains control information used by the restore control information RCI directive to reset the transmit channel and receive channel CIBs to default (and compatible) values.

The following are the BLIM Diagnostic default CIB values:

CIB Byte	Value	Description
MODE(Byte 0):	$1C_{16}$	VRC Parity - Odd VRC Parity - Enabled (ON) Character Length - 8 Bits/ Character
RATE(Byte 1):	$7F_{16}$	Baud Rate - 9600
FRMT(Byte 2):	00_{16}	Ring Detection - Disabled (OFF) Code Converter - Disabled (OFF) Line Control - Private Block Check Character - CRC-16
DLES(Byte 4):	10_{16}	DLE Character (CTRL-P)
SYNC(Byte 5):	16_{16}	SYN Character (CTRL-V)
ACK0(Byte 6):	30_{16}	ACK0 Character (CTRL-P,0)
ACK1(Byte 8):	31_{16}	ACK1 Character (CTRL-P,1)
WACK(Byte 10):	$3B_{16}$	WACK Character (CTRL-P,;)
RVI(Byte 12):	$3C_{16}$	RVI Character (CTRL-P,)
DLE(Byte 14):	10_{16}	DLE Character (CTRL-P)
ENQ(Byte 16):	05_{16}	ENQ Character (CTRL-E)
EOT(Byte 18):	04_{16}	EOT Character (CTRL-D)
ETB(Byte 20):	17_{16}	ETB Character (CTRL-W)
ETX(Byte 22):	03_{16}	ETX Character (CTRL-C)
ITB(Byte 24):	$1F_{16}$	ITB Character (CTRL-_)
NAK(Byte 26):	15_{16}	NAK Character (CTRL-U)
SOH(Byte 28):	01_{16}	SOH Character (CTRL-A)
STX(Byte 30):	02_{16}	STX Character (CTRL-B)
SYN(Byte 32):	16_{16}	SYN Character (CTRL-V)

The program explicitly writes control information to all assigned GPDC BLIM channels if it executes a Write Control Information (WCI) directive. The program also implicitly writes control information to a GPDC BLIM channel prior to issuing any Start I/O instruction, if the following conditions occur:

- The program has not previously written control information to the channel.
- There is a change in the contents of the channel's CIB since the CIB was last written.
- The Gould Diagnostic Executive Program (DEXP) is reinitialized by entering RST and RUN on the operator's console.

6.8 Control Information Block (CIB) Directives

Use the following directives to modify the contents of the transmit, receive, or default CIBs:

TCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the transmit channel CIB.

RCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the receive channel CIB.

CCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of both the transmit channel and receive channel CIBs.

DCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the default, transmit channel and receive channel CIBs.

For more information, refer to the individual unique directives.

6.9 Control Information Block (CIB) Directive Arguments

Use the following arguments with CIB directives:

BCC(LRC-8)

Conditions the block check character field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB is the receive channel CIB) an eight-bit longitudinal redundancy check (LRC) character. Although the program generates an eight-bit block check character, the program transmits the character according to the specified character length. Therefore, select LRC-8 for six, seven, and eight-bit transmissions. Always select LRC-8 for seven-bit transmissions.

BCC(CRC-12)

Conditions the block check character field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB is the receive channel CIB) a 12-bit cyclic redundancy check (CRC) character. Select CRC-12 for six-bit transmissions.

BCC(CRC-16)

Conditions the block check character field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB is the receive channel CIB) a 16-bit cyclic redundancy check (CRC) character. Select CRC-16 for eight-bit transmissions.

BCC(CRC-CCITT)

Conditions the block check character field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB specified is the receive channel CIB) a 16-bit cyclic redundancy check (CRC) character as defined by the Consultative Committee on International Telegraphy and Telephony (CCITT).

BR(n)

Conditions the baud rate field in the specified CIB as follows:

(n)	Baud Rate	Maximum Transfer Length (Bytes)
150	150 bps	256
300	300 bps	256
600	600 bps	256
900	900 bps	256
1200	1200 bps	512
1800	1800 bps	768
2400	2400 bps	1024
3600	3600 bps	1280
4800	4800 bps	1536
7200	7200 bps	1792
9600	9600 bps	2048
14400	14.4 kbps	2048
19200	19.2 kbps	2048
28800	28.8 kbps	2048
38400	38.4 kbps	2048

The program dynamically changes the maximum transfer length in an attempt to maintain the same transfer time for all baud rates. In nearly all cases, the maximum transfer length is sufficient to allow the controller's one-second timer to expire, forcing the program to automatically insert SYN or DLE-SYN characters to maintain synchronization. The program chooses the maximum allowable transfer length (20488) to preclude the possibility of generating an overflow in the block check character generator.

CC(OFF)

Conditions the code converter field in the specified CIB so that the converter is disabled (OFF).

CC(ON)

Conditions the code converter field in the specified CIB so that the converter is enabled (ON).

CL(n)

Conditions the character length field in the specified CIB as follows:

(n)	Character Length
6	6 data bits/character
7	7 data bits/character
8	8 data bits/character

LC(PRIVATE)

Conditions the line control field in the specified CIB to indicate that the communications link is private (dedicated to the channel) and need not be connected prior to a transmission. Line control automatically defaults to private in response to a jumper half-duplex (JH) device assignment.

LC(SWITCHED)

Conditions the line control field in the specified CIB to indicate that the communications link is switched (not dedicated to the channel) and must be connected prior to a transmission. Line control automatically defaults to switched in response to a modem half-duplex (MH) device assignment.

PAR(OFF)

Conditions the vertical parity field in the specified CIB so that the program does not generate (if the specified CIB is the transmit channel CIB) or check (if the specified CIB is the receive channel CIB) vertical parity.

PAR(ON,ODD)

Conditions the vertical parity field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB is the receive channel CIB) odd vertical parity.

PAR(ON,EVEN)

Conditions the vertical parity field in the specified CIB so that the program generates (if the specified CIB is the transmit channel CIB) or checks (if the specified CIB is the receive channel CIB) even vertical parity.

The effect of selecting vertical parity generation/checking for an n data bits/character transmission is to transfer n+1 data bits/character. Since the transmission is synchronous serial-by-bit, the frame length must be identical for both channels (i.e., both channels must have vertical parity selected or not selected) to avoid loss of synchronization.

RING(OFF)

Conditions the ring interrupt detection field in the specified CIB so that the ring interrupt is disabled (OFF).

RING(ON)

Conditions the ring interrupt detection field in the specified CIB so that the ring interrupt is enabled (ON).

For jumpered half-duplex channel configurations, the program uses the IOCL order Transmit Ring to generate a ring detected service interrupt (SI). For modem half-duplex channel configurations, the program generates the ring detected service interrupt (SI) during the dial process. Provide a delay following the receipt of the ring detected service interrupt and the issuance of the IOCL order Receive Data to allow the modems time to synchronize. This delay is typically two seconds.

CHAPTER 7

I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview

The Gould GPDC Binary Synchronous Line Interface Module Diagnostic contains optional capabilities that provide comprehensive listings of I/O activity during program execution.

These capabilities include I/O Command Trace and Data Recorder.

By setting the appropriate options, these routines produce a visual representation to the listed output (LO) file of the data being manipulated and I/O commands being issued.

Use these listings to examine in greater detail, the processor activity by supplementing error messages and verifying that the actual data being generated coincides with the commands being issued.

7.2 I/O Command Trace

The I/O Command Trace provides a record of all I/O instructions issued and in the order the program issues them during program execution. Option 17 enables the I/O Command Trace and sends the information to the LO file.

7.2.1 Starting Execution

To begin I/O Command Trace execution, set Option 17 (enables Command Trace) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the LO file:

```
<<<<'taskname'----hh:mm:ss>>>>
TST:ttt, DEV:aa-c-00, PC:ppppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal GPMC IOM Controller address.
c	specifies the one-character hexadecimal BLIM channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.2.2 Milestone Messages

The program writes the following milestone messages to the listed output (LO) file during an I/O Command Trace:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx,  IOCLQP:00qqqqqq
```

xxxxxxxx specifies the Command Device (CD) instruction issued.

qqqqqq specifies the address of the IOCL queue if the instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the value.

The IOCL queue contents (repeated for all IOCDs within the IOCL queue):

```
IOCD(ii):ooffcccc 00bbbbbb
```

ii specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo specifies the contents of the order byte in IOCD Word 1.

ff specifies the contents of the flag byte in IOCD Word 1.

cccc specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb specifies the hexadecimal initial buffer address in IOCD Word 2.

7.2.3 Terminating the I/O Command Trace

To terminate the I/O Command Trace, issue an Attention command and reset Option 17 (enables Command Trace) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 17
```

7.3 Data Recorder

The Data Recorder provides a dump of the actual data being manipulated during program execution. The dump shows how the data appears during I/O activity. Option 18 enables Data Recorder and Option 9 enables Data Recorder on error. With either option set, the routine sends a dump of data to the listed output (LO) file.

7.3.1 Starting Execution

To begin Data Recorder execution, set Option 18 (enables Data Recorder) or Option 9 (enables Data Recorder on error) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the LO file:

```
<<<<'taskname'----hh:mm:ss>>>>
TST:ttt, DEV:aa-c-00, PC:ppppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal GPMC IOM Controller address.
c	specifies the one-character hexadecimal BLIM channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.3.2 Milestone Messages

The program writes the following milestone messages during the Data Recorder routine:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx, IOCLQP:00qqqqqq
```

xxxxxxxx	specifies the Command Device (CD) instruction issued.
qqqqqq	specifies the address of the IOCL queue if the instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the address.

The I/O command doubleword (IOCD) specification (for each I/O Command List (IOCD) in the IOCL queue):

```
IOCD (ADDR)   (WRD1)    (WRD2)    ORDER IS:  order
(ii) qqqqqq   ooffcccc  00bbbbbb  FLAGS ARE: flag
                                              .
                                              .
                                              .
                                             flag
```

ii — specifies the queue slot number of the IOCD in the IOCL queue.

qqqqqq — specifies the address of the IOCD in the IOCL queue.

oo — specifies the contents of the order byte in IOCD Word 1.

ff — specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb — specifies the hexadecimal initial buffer address in IOCD Word 2.

order — specifies the IOCD order. If the listed output (LO) file assignment is to a console device, the program writes the hexadecimal value of the order byte. Otherwise, it writes a text description of the order:

oo = 00	STOP
01	TRANSMIT DATA
02	RECEIVE DATA
03	CONNECT COMMUNICATIONS CHANNEL
04	WRITE CONTROL INFORMATION
05	DISCONNECT COMMUNICATIONS CHANNEL
06	RING(OUT)
07	TRANSFER-IN-CHANNEL
08	ILLEGAL

flag — specifies the IOCD flags. If the LO file assignment is to a console device, the program writes the hexadecimal value of the flag byte. Otherwise, it writes a text description of each set flag bit:

ff = 40	INTERRUPT ON ZERO BYTE COUNT
20	HALT ON TRANSMISSION ERROR
10	SUPPRESS INCORRECT LENGTH
08	INTERRUPT ON UNUSUAL END
04	COMMAND CHAIN
02	DATA CHAIN
01	INTERRUPT ON CHANNEL END

The message for each I/O command doubleword (IOCD) involving a data transfer:

```
IOCD (ADDR)   DATA LENGTH SPECIFIED:  ccccc TRANSFERRED: ttttt
(ii) bbbbbb   mmm mmm a a a . .  a a mmm mmm...
     (Rnnnnn) xx..xx..xxxxxx.....xxxxxx..xx...
```

ccccc — specifies the decimal value of the initial byte count in IOCD Word 1.

ttttt — specifies the number of decimal bytes actual transferred (only written for the last executed IOCD in the I/O Command List (IOCL) queue).

ii — specifies the queue slot number of the IOCD in the IOCL queue.

bbbbbb — specifies the hexadecimal initial buffer address in IOCD Word 2.

Rnnnnn — specifies the relative frame number of the first data byte in the line.

mmm — specifies the mnemonic name of an ASCII control character associated with the hexadecimal representation of the ASCII or EBCDIC data byte ..xx.. .

a — specifies the graphic character associated with the hexadecimal representation of the ASCII or EBDIC data byte ..xx.. .

..xx.. — specifies the hexadecimal representation of an ASCII or EBCDIC data byte.

The message for the terminal IOCD:

```
IOCD ADDR: DEVICE STATUS AT TERMINAL IOCL:  status
(ii) qqqqqq
```

ii — specifies the queue slot number of the last executed IOCD in the IOCL queue.

qqqqqq — specifies the address of the last executed IOCD in the IOCL queue.

status — specifies the test device (TDV) status at I/O completion. If the value of the TDV status is 0000_{16}, the program writes one of the following messages:

I/O COMPLETE	Normal completion.
I/O TIME-OUT	I/O not completed within expected time, HALT I/O issued to terminate operation.

If the value of the test device (TDV) status is other than 0000_{16} and the listed output (LO) file assignment is a console device, the program writes the hexadecimal value of the TDV status. Otherwise, it writes a text description of each set TDV status bit as follows:

8... LINK DISCONNECT/DATA SET NOT READY
4... BCC ERROR
2... FORMAT ERROR
1... RECEIVER OVERRUN
.8.. VRC ERROR
.4.. (RX) RING DETECTED/(TX) NOT C2S
.2.. LOSS OF SYNC
.1.. CHANNEL INOPERABLE/NOT INITIALIZED
..8. N/A
..4. N/A
..2. TRANSMISSION ERROR
..1. INCORRECT LENGTH
...8 UNUSUAL END
...4 ILLEGAL ORDER
...2 INTERRUPT PENDING
...1 CHANNEL END

7.3.3 Terminating Data Recorder

To terminate the Data Recorder routine, issue an Attention command and reset Option 18 (enables Data Recorder) or Option 9 (enables Data Recorder on error) using the DEXP Reset command.

Example (if Option 18 is set)

```
@@A
DXP>RESET taskname,OPT 18
```

Gould GPDC Synchronous Line Interface Module

(GP.SLM.r)

Diagnostic Description

Diagnostic Number: 001032-985r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005760-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould GPDC Synchronous Line Interface Module (GP.SLM.r) Diagnostic Description Release 1.0, Publication Order Number **326-005760-000,** was printed May 1985.

This manual contains the following pages:

Title Page
Copyright page
iii/iv through vii/viii
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-2
4-1 through 4-4
5-1 through 5-6
6-1 through 6-20
7-1 through 7-6

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Model Number Assignments 2-2
2.3 Program Option Set 2-2

CHAPTER 3 PROGRAM EXECUTION

3.1 Starting Program Execution 3-1
3.2 Milestone Messages 3-1
3.3 Terminating Program Execution 3-2

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Message Routines 4-1
4.2 Unique Error Messages 4-3

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Link Disconnected Status Test 5-1
5.3 Test 2 Channel Inoperable Status Verification 5-2
5.4 Test 3 Illegal Order Test 5-2
5.5 Test 4 Transmit and Receive All Zeros 5-2
5.6 Test 5 Transmit and Receive All Ones 5-2
5.7 Test 6 Transmit and Receive Floating Ones 5-2
5.8 Test 7 Transmit and Receive Floating Zeroes 5-2
5.9 Test 8 Transmit and Receive Binary Progression 5-2
5.10 Test 9 Transmit and Receive Random Data 5-2
5.11 Test 10 Transmit and Receive Alternating Bit Pattern: AA55AA55 5-2

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould GPDC Synchronous Line Interface Module Diagnostic (GP.SLM.r) is a level two macrodiagnostic program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 38 tests, which verify the correct operation of GPDC instructions. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring operator intervention by setting the appropriate options.

During execution, the program writes milestone messages and error message sequences to the listed output (LO) file which the program assigns by default to the operator's console.

The DEXP's directive language compiler and directives supplied by this program provide:

- the ability to create new directive test sequences
- the ability to reorganize test sequences

The program reads operator-composed directive sequences from the command control (CC) file (if in Batch mode) or from the operator communications (OC) file (if in Interactive mode). The CC file assignments are to disc or tape and the OC file assignments are to the operator's console or a terminal configured on an eight-line asynchronous controller.

1.2 Resource Requirements

The Gould GPDC Synchronous Line Interface Module Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory
- One of the following:

  Model 9103 GPMC IOM/Controller (16 megabyte)
  Model 9104 GPMC IOM/Controller (512 kilobyte)
  Model 9109 GPDC SLIM Controller

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

The Gould GPDC Synchronous Line Interface Module (SLIM) Diagnostic (GP.SLM.r) provides the capability of testing from two to 16 dual sync channels connected in Half-Duplex configuration or from four to 16 channels connected in Full-Duplex configuration.

The Gould SLIM Diagnostic is a self-initializing program and sets up its default values internally. Therefore, it issues no initialization prompts after loading. To load the program, follow the loading procedures in the appropriate Diagnostic Facility User's Manual. The program name for loading the Gould SLIM Diagnostic is GP.SLM.r.

The Gould SLIM Diagnostic sets up the following default assignments:

- A jumpered Half-Duplex configuration
- GPMC Controller address of 20
- A GPDC SLIM channel address of 00 jumpered to 01

The SLIM Diagnostic defaults to a jumpered Half-Duplex configuration with a GPMC Controller address of 20_{16} and GPDC SLIM channel address specifying channel 00_{16} jumpered to channel 01_{16}.

To modify the default configuration, the GPMC IOM/Controller address, and/or the GPDC SLIM channel addresses, use the DEXP Assign command with the following device type codes:

Code	Description
JH	Jumpered Half-Duplex
JF	Jumpered Full-Duplex

Use the directives APR, APT, ASR, and AST to assign the primary receive and transmit channels and also the secondary receive and transmit channels, respectively. The following table defines the channel numbers assigned by these directives for each of the configurations:

Device Type Code	Primary		Secondary	
	Receive (APR)	Transmit (APT)	Receive (ASR)	Transmit (AST)
JH	c_0	c_1	--	--
JF	c_0	c_1	c_2	c_3

c_0 specifies the primary receive channel.
c_1 specifies the primary transmit channel.

c_2 specifies the secondary receive channel.
c_3 specifies the secondary transmit channel.

n Full-Duplex operation, c_1 is equal to c_0+1 and c_3 is equal to c_2+1. Connect the primary receive channel (c_0) to the secondary transmit channel (c_3) and the primary transmit channel (c_1) to the secondary receive channel (c_2).

2.2 Model Number Assignments

The Gould GPDC Synchronous Line Interface Module (SLIM) Diagnostic sets up the default model number assignment as 9103 (a GPMC IOM Controller capable of addressing 16 megabytes of memory). To change the model number assignment to 9104 (a GPMC IOM Controller, capable of addressing 512K bytes of memory), use the MOD directive.

2.3 Program Option Set

Once loaded, the Gould GPDC Synchronous Line Interface Module Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould GPDC Synchronous Line Interface Module Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the bell on the operator's console or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR...; [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option.

Option Default Effect If Set

Option	Default	Effect If Set
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Reset	Runs directive tests only. Setting this option runs only tests coded in directive language.
8	Reset	Inhibits operator intervention tests. Setting this option inhibits any test requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count message to the listed output (LO) file.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone message to the listed output (LO) file.
12	Reset	Inhibits error message expansion.
13	Reset	Inhibits test device (TDV) status error messages.
14	Reset	Inhibits test buffer address (TBA) status error messages.
15	Reset	Inhibits data error messages.
16	Reset	Inhibits test command list address (TCLA) error messages.
17	Reset	Enables Command Trace.
18	Reset	Enables Data Recorder.

CHAPTER 3

PROGRAM EXECUTION

3.1 Starting Program Execution

After setting options, issue a DEXP Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-985r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messsages to the listed output (LO) file.

3.2 Milestone Messages

The program verifies the device assignments made and, if the device type code assigned is other than JH or JF, the program writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED:  INVALID DEVICE TYPE CODE tt
```

Reassign the device to be tested using one of the valid device type codes.

The program then verifies the presence of the GPMC IOM Controller and, if it is absent, the program writes the following message to the OC file:

```
'taskname' TERMINATED:  GPMC:aa ABSENT
```

Reassign the device to be tested using the correct GPMC IOM Controller address.

The program then verifies that only even GPDC Controller addresses are assigned in a Full-Duplex mode. If an odd GPDC controller address is specified, the program writes the following message to the OC file:

```
'taskname' TERMINATED:  GPDC ASSIGN ERROR
```

Reassign the channels to be tested specifying only the even GPDC Controller addresses.

The program then verifies the presence of the Synchronous Line Interface Module (SLIM) Controllers and, if absent, it writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED:  GPDC:c ABSENT
```

Reassign the device to be tested using the correct GPDC Controller address.

The program then sequentially executes all designated tests for each GPDC Controller assigned.

If Option 11 (writes milestone messages) is set, the program writes the following message to the listed output (LO) file prior to the execution of each test:

```
hh:mm:ss 'taskname' TEST: n
```

If Option 10 (writes pass count messages) is set, the program writes the following message to the LO file after completing all designed tests:

```
hh:mm:ss 'taskname' FINISHED PASS: n
```

3.3 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Message Routines

The Gould GPDC Synchronous Line Interface Module Diagnostic verifies the correct operation of the controller (I/O command) function through a fixed sequence of checks. If the program detects an error and Option 0 (inhibits error messages) is reset, it writes a message describing actual and expected results during execution of an I/O command.

The program writes all error messages to the listed output (LO) file which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

   ```
   <<<<'taskname '----hh:mm:ss>>>>
   ```

   taskname — specifies the one to eight-character program name indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

   ```
   TST:ttt,DEV:aa-c-00,PC:pppp
   ```

   ttt — specifies the test number of the erring test.

   aa — specifies the two-character hexadecimal GPMC Controller address.

   c — specifies the one-character hexadecimal SLIM channel address.

   pppp — specifies the program counter value within the program.

3. The erring directive (for directive tests only)

   ```
   DRTV:directive
   ```

   directive — specifies the erring directive.

4. The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable)

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx specifies the Command Device (CD) instruction issued.

qqqqqq specifies the address of the IOCL queue if the CD instruction is a Start I/O (SIO) instruction. Otherwise, the program omits the value.

5. The IOCL queue contents

```
IOCD(ii):ooffcccc 00bbbbbb
```

ii specifies the IOCL queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.

oo specifies the contents of the order byte in IOCD Word 1.

ff specifies the contents of the flag byte in IOCD Word 1.

cccc specifies the hexadecimal value of the initial byte count in IOCD Word 1.

bbbbbb specifies the initial buffer address in IOCD Word 2.

4.2 Unique Error Messages

```
IOCD(ii)STATUS
EXP(DV):ddddcccc   (BA):00bbbbbb   (CLA):00aaaaaa
ACT(DV):ddddcccc   (BA):00bbbbbb   (CLA):00aaaaaa
```

ii — specifies the queue slot number of the IOCD in the IOCL queue where the status error occurred.

EXP — identifies the expected status values.

ACT — identifies the actual status values.

dddd — specifies the hexadecimal value of the status.

cccc — specifies the hexadecimal value of the remaining byte count.

bbbbbb — specifies the hexadecimal value of the test buffer address (TBA) status.

aaaaaa — specifies the hexadecimal value of the test command list address (TCLA) status.

If Option 12 (inhibits error message expansion) is reset, the program writes one of the following messages for each status bit using a plus sign to indicate an unexpected status bit and a minus sign to indicate a missing status bit in either the actual or expected status reports.

```
8...  LINK DISCONNECTED
4...  CRC ERROR
2...  FORMAT ERROR
1...  OVERFLOW/UNDERFLOW
.8..  VRC ERROR
.4..  RING DETECTED
.2..  LOSS OF SYNC
.1..  CHANNEL INOP/NOT INITIALIZED
..8.  N/A
..4.  CARRIER OFF
..2.  TRANSMISSION ERROR
..1.  INCORRECT LENGTH
...8  UNUSUAL END
...4  ILLEGAL ORDER
...2  INTERRUPT PENDING
...1  CHANNEL END
```

If the program executes the specified Command Device (CD) instruction and receives no service interrupt (SI), it writes the following message:

```
I/O TIMEOUT (NO SI)
```

If data comparison discrepancies occur, the program writes the following message:

```
IOCD(ii):DATA ERROR(S)
FR#:    EXP: ACT:
ffff     dd    dd
 .       .     .
 .       .     .
 .       .     .
ffff     dd    dd
xxxx OUT OF yyyy
```

ii — specifies the queue slot number of the I/O command doubleword (IOCD) in the I/O Command List (IOCL) queue where the data errors occurred.

FR# — identifies the frame numbers in error.

EXP — identifies the expected input buffer value.

ACT — identifies the actual input buffer value.

ffff — specifies the frame number in error relative to the first frame of the IOCD where the error occurred (a maximum of five error frames are reported for each IOCD).

dd — specifies the hexadecimal value of the expected or actual data element.

xxxx — specifies the total number of frames in error for the IOCD.

yyyy — specifies the total number of frames compared for the IOCD.

The total number of frames compared by the program is equal to the number of bytes specified in the initial byte count for each IOCD. However, for the terminal IOCD, if the test device remaining byte count indicates that not all bytes requesting transfer are transferred, the program compares only those bytes actually transferred.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould GPDC Synchronous Line Interface Module (SLIM) Diagnostic contains 39 tests which run in a continuous sequence or individually by operator request. The test sequences verify the correct functioning of GPDC Controller orders.

5.2 Test 1 Link Disconnected Status Test

This test verifies that the SLIM detects and reports a link disconnected condition.

The program writes the following message to the operator communications (OC) file and suspends program execution:

```
'taskname' SUSPENDED:DISCONNECT LINK DEV:aa-c
```

Disconnect the cable connected to the SLIM channel specified by GPMC address aa and GPDC address c and continue program execution by issuing a DEXP Continue command.

The program writes the following message to the OC file:

```
'taskname' RECONNECT LINK DEV:aa-c
```

Reconnect the cable for the GPDC previously disconnected.

The program repeats this test for all of the assigned devices on the first pass of the program's execution and then inhibits the test until a new Assign command is entered through the command control (CC) file.

After the last device is tested, the program writes following message to the OC file:

```
'taskname' SUSPENDED:
```

Reconnect the cable for the device at this time, then issue a DEXP Continue command to resume execution.

The program skips this test if Option 7 (runs directive tests only) or Option 8 (inhibits operator intervention tests) is set.

5.3 Test 2 Channel Inoperable Status Verification

This test verifies that the SLIM detects and reports a Channel Inoperable/Not Initialized condition.

The program skips this test if Option 7 (runs directive tests only) is set.

5.4 Test 3 Illegal Order Test

This test verifies that the SLIM detects illegal I/O Command List (IOCL) orders and posts the correct test device (TDV) status.

The program skips this test if Option 7 (runs directive tests only) is set.

5.5 Test 4 Transmit and Receive All Zeros

This test verifies that the SLIM transmits and receives an all zeros pattern. It is used primarily to detect picked bits.

5.6 Test 5 Transmit and Receive All Ones

This test verifies that the SLIM transmits and receives an all ones pattern. It is used primarily to detect dropped bits.

5.7 Test 6 Transmit and Receive Floating Ones

This test verifies that the SLIM transmits and receives a one bit in each bit position of the data byte.

5.8 Test 7 Transmit and Receive Floating Zeroes

This test verifies that the SLIM transmits and receives a zero bit in each bit position of the data byte.

5.9 Test 8 Transmit and Receive Binary Progression

This test verifies that the SLIM transmits and receives each eight-bit binary data pattern.

5.10 Test 9 Transmit and Receive Random Data

This test verifies that the SLIM transmits and receives a random data pattern.

5.11 Test 10 Transmit and Receive Alternating Bit Pattern: AA55AA55

This test verifies that the SLIM transmits and receives an alternating bit pattern.

5.12 Test 11 Transmit and Receive Varying Character Length and Parity

This test verifies that the SLIM transmits and receives data with each of the character length/parity combinations. The test contains the following eight subtests, one for each combination as follows:

Subtest	Character Length/Parity
A	6 Bits/Even Parity
B	6 Bits/Odd Parity
C	7 Bits/No Parity
D	8 Bits/No Parity
E	7 Bits/Even Parity
F	7 Bits/Odd Parity
G	8 Bits/Even Parity
H	8 Bits/Odd Parity

5.13 Test 12 Terminate Transmit on Character Recognition

This test verifies that the SLIM terminates a transmit operation upon recognizing a specified character.

5.14 Test 13 Terminate Receive on Character Recognition

This test verifies that the SLIM terminates a receive operation upon recognizing a specified character.

5.15 Test 14 Transmit and Receive Varying Record Lengths: 1,2,4,8,...,PV/2 Bytes

This test verifies the ability to transmit and receive records of varying lengths.

5.16 Test 15 Remaining Byte Count Status Test

This test verifies the ability to correctly return the remaining byte count in the test device (TDV) status.

5.17 Test 16 Strip Character in Transmit Operation

This test verifies that the SLIM recognizes and strips a specified character in a transmit operation.

5.18 Test 17 Strip Character in Receive Operation

This test verifies that the SLIM recognizes and strips a specified character in a receive operation.

5.19 Test 18 Block Check Code Verification

This test verifies the ability to perform the reset block check character and the perform block check functions.

5.20 Test 19 VRC Error Detection

This test verifies that the SLIM detects and reports a vertical redundancy check (VRC) error condition.

5.21 Test 20 CRC Error Detection

This test verifies that the SLIM detects and reports a cyclic redundancy check (CRC) error condition with each of the block check codes.

5.22 Test 21 Sync Pattern Recognition

This test verifies that the SLIM transmits and recognizes the receipt of three different sync character patterns and verifies the correct functioning of the Strip Sync Characters on receive mode bit.

5.23 Test 22 Set Action Counter Function

This test verifies that the SLIM sets each bit position of the action counter and performs a terminate function on the perform function on action counter compare condition.

5.24 Test 23 Clear Action Counter Function

This test verifies that the SLIM clears each bit position of the action counter.

5.25 Test 24 Add Action Counter Function

This test contains four subtests to verify that the SLIM correctly performs the add to action counter function.

5.26 Test 25 Multiple Condition/Function Test

This test verifies that the SLIM controller correctly processes multiple condition/function triplets for recognition character entries.

5.27 Test 26 Set and Reset Transparent Mode

This test verifies that the SLIM correctly performs the Set Transparent and Reset Transparent functions in both transmit and receive modes.

5.28 Test 27 Insert DLE Character in Transmit Transparent Mode

This test verifies that the SLIM inserts a Data Link Escape (DLE) character in transmit transparent mode following a recognition character setting the action counter to a value of seven unless a zero byte count condition accompanies the character.

5.29 Test 28 Insert DLE Character on Receive

This test verifies that the SLIM inserts a DLE character in the Receive function.

5.30 Test 29 Transmit Ring, Receive Ring Interrupt

This test verifies that the SLIM transmit channels perform a ring out operation and the receive channels detect and report a ring detected status.

5.31 Test 30 Send and Test One-Second Sync

This test verifies that the SLIM sends and tests the one-second sync patterns required for binary synchronous communications. It contains two subtests, one for nontransparent mode and one for transparent mode.

5.32 Test 31 Loss of Sync Status Test

This test verifies that the SLIM receive channel detects and reports a loss of sync condition. It contains two subtests, one for nontransparent mode and one for transparent mode.

The program skips this test if Option 7 (runs directive tests only) is set.

5.33 Test 32 Binary Synchronous Communications: Normal Text Block

This test verifies that the SLIM transmits and receives the binary synchronous communications (BISYNC) normal text block consisting of a Start-of-Text (STX) character, the text string, and an End-of-Text character (ETX).

5.34 Test 33 Binary Synchronous Communications: Heading, Normal Text Block

This test verifies that the SLIM transmits and receives the BISYNC message, consisting of a heading followed by a normal text block. The heading consists of a Start-of-Heading (SOH) character and a Block-of-Heading data.

5.35 Test 34 Binary Synchronous Communications: Intermediate Transmission Block

This test verifies that the SLIM transmits and receives a BISYNC message consisting of an intermediate transmission block. The intermediate transmission block consists of a Start-of-Text (STX) character, the text string, and an End-of-Intermediate-Transmission-Block (ITB) character.

5.36 Test 35 Binary Synchronous Communications: Transparent Data

This test verifies that the SLIM transmits and receives a BISYNC transparent data block. The transparent data block consists of a Data Link escape (DLE) character, a Start-of-Text (STX) character, the transparent data, and an End-of-Text (ETX) character.

5.37 Test 36 Binary Synchronous Communications: Last Text Block

This test verifies that the SLIM transmits and receives the BISYNC last text block consisting of a Start-of-Text (STX) character, a text string, and an End-of-Transmission-Block (ETB) character.

5.38 Test 37 Binary Synchronous Communications: Polling Sequence

This test verifies that the SLIM transmits the BISYNC polling sequence consisting of an End-of-Transmission (EOT) character, a pad character, two Synchronous Idle (SYN) characters, the polling address, and an Enquiry (ENQ) character.

5.39 Test 38 Binary Synchronous Communications: Data Link Control

This test contains eight subtests to verify that the SLIM transmits and receives data link control sequences as follows:

Subtest	Control Sequence
A	ACK0 - Affirmative Acknowledgment
B	ACK1 - Affirmative Acknowledgment
C	NAK - Negative Acknowledgment
D	WACK - Wait-Before-Transmit Positive Acknowledgment
E	RVI - Reverse Interrupt
F	TTD - Temporary Text Delay
G	EOT - End-of-Transmission
H	EVQ - Enquiry

CHAPTER 6

DIRECTIVE LANGUAGE

6.1 Directive Overview

The Gould GPDC Synchronous Line Interface Module Diagnostic offers the capability of creating new tests or reorganizing existing tests through directive language.

The Diagnostic Executive Program contains a Directive Language Compiler and a set of standard directives available to all level two macrodiagnostic programs. In addition, each program contains unique directives to perform functions specific to the program.

Combining directives to create tests helps to examine the hardware in greater detail or with different emphasis to isolate possible problems on boards or board sets.

The following sections explain how to use these directive language components:

- directives
- parameters
- arguments

6.2 Using Directives

Directives are mnemonics which represent functions. Use directives to create tests that perform oprations which help isolate problems on boards or board sets.

For information on creating operator-composed directive tests, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A program must be loaded, but not executing, in order to run operator-composed directive tests. If a program is running and a directive string is entered, DEXP issues the following prompt:

'taskname' ACTIVE [C,T]?

Respond with C to continue the directive string and terminate the program, or with T to terminate the directive string and continue the program.

Use directives as follows:

In Interactive mode

DXP>taskname LA,CR,DO <cr>

Enter directives and press the Return key in response to a DXP> prompt. DEXP then performs the directive functions and continues execution.

In Batch Mode

DXP>BATCH bfile

bfile specifies a command file.

Create a command file containing directives, then use the Batch command to enter Batch mode and execute the file. (For more information on command files, refer to the Diagnostic Facility User's Manual.)

In a Macro

DXP>taskname MAC,mfile

mfile specifies a macro name.

Create a macro containing directives using the Define command, then execute the macro. (For more information on macros, refer to commands in the DEXP Diagnostic Description.)

6.3 Using Parameters

Parameters are memory words which contain data used in program execution. Each level two macrodiagnostic program contains 26 parameters which are either reserved for internal program use or available for general use. Use the P directive to modify general use parameters.

Example

DXP>PK*L

If parameter K contains a value of 50, and parameter L contains a value of 2, then PK*L multiplies the value in parameter K (50) by the value in parameter L (2) and stores the result (100) in parameter K. If another PK*L executes now that the value in parameter K is 100, the mathematical equivalent is 100 x 2 = 200, with parameter K storing the result (200).

Use other operand values such as addition or subtraction for incrementing or decrementing parameters.

A list of valid operand values follows:

=	replacement	.O.	Logical OR
+	addition	.E.	logical EXCLUSIVE OR
-	subtraction	.N.	two's complement
*	multiplication	.C.	one's complement
/	division	>	circular shift right
.A.	logical AND	<	circular shift left

A list of Gould GPDC Synchronous Line Interface Module Diagnostic parameters and a summary of their functions follow:

Parameter	Use	Description
A	Reserved	Contains the input/output buffer address attributes (F=1,C=00).
B	Reserved	Contains the baud rate used for I/O transfer time calculation (default 4800 baud).
C	Available	Available for general use.
D	Reserved	Contains the current output buffer data pattern if the output buffer is set using a PD directive.
E	Available	Available for general use.
F	Reserved	Contains the program control flags. Bit 31=0: Half-Duplex Bit 31=1: Full-Duplex
G	Available	Available for general use.
H	Available	Available for general use.
I	Reserved	Contains the current output buffer word one data pattern if the output buffer word one is set using a PI directive.
J	Available	Available for general use.
K	Available	Available for general use.
L	Available	Available for general use.
M	Reserved	Contains the test device (TDV) status mask pattern: B00 through 5: TDV Device Status B16 through 31: TDV Remaining Byte Count (RBC)
N	Available	Available for general use.
O	Available	Available for general use.
P	Available	Available for general use.
Q	Available	Available for general use.
R	Available	Available for general use.
S	Reserved	Contains a number between 1 and 16 (decimal) corresponding to a letter from A through Z. It is used by the ST directive as a subtest designator.

Parameter	Use	Description
T	Available	Available for general use.
U	Available	Available for general use.
V	Reserved	Contains the default transfer count (in bytes).
W	Reserved	Contains the transfer count for W, H, and HS directives.
X	Reserved	Contains the hold duration for H and HS directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for JZ directives.

6.4 Using Arguments

Use optional arguments (in any order) to extend or modify the function of a directive.

Example

```
@@A
DXP>LA3
```

The argument (3) modifies the directive LA (loads ASCII set) to permit three bytes of the diagnostic's output buffer to be loaded with ASCII instead of the default number of bytes specified in parameter W. (For more information on the LA directive, refer to the Directive Language chapter in the DEXP Diagnostic Description.)

Each directive section contains a summary of arguments pertaining to that directive. However, not all directives have associated arguments.

6.5 DEXP Directive Summary

The Gould Diagnostic Executive Program contains nine directive categories which are available to all level two macrodiagnostic programs. For more information on standard directives, refer to the Directive Language chapter in the DEXP Diagnostic Description.

A list of standard directives and a summary of their functions follow:

"A" Directives

A	Assigns a channel address and device subaddress and creates CD and TD instructions.
ACAn	Assigns a channel address and creates CD and TD instructions.
ADSAn AUn	Assigns a device subaddress and creates CD and TD instructions.

"C" Directives

CLn	Cycles output buffer left.
CLp	Cycles output buffer left by the value in the parameter specified by p.
CRn	Cycles output buffer right.
CRp	Cycles output buffer right by the value in the parameter specified by p.

"D" Directives

DIn	Dumps the input buffer.
DOn	Dumps the output buffer.
DP	Dumps the contents of all parameters.
DPp	Dumps the contents of the parameter specified by p.

"H" Directives

Hn	Holds program execution for the specified number of system clock ticks.
HSn	Holds program execution for the specified number of seconds.

"J" Directives

The J (jump) directives perform as stated in the following syntax:

Jn

JZ

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k

J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2

n	specifies statement number.
Z	indicates statement number in parameter Z.
k	specifies a number (1 through 32,767).
p	specifies the parameter (A through Z).
(op)	specifies legal jump operands.
/	identifies the delimiter between the jump and the expression which describes the number of jumps to be performed.

"L" Directives

L='' or L( )	Loads the output buffer with 'any literal value'.
LAn	Loads the output buffer with the 64-character printable ASCII subset.
LBKn	Loads the output buffer with ASCII blanks.
LBPn	Loads the output buffer with a binary progression.
LBRn	Loads the output buffer with a binary regression.
LFOn	Loads the output buffer with a floating one pattern.
LFZn	Loads the output buffer with a floating zero pattern.
LIOn	Loads the input buffer with the contents of the output buffer.
LIRn	Loads the output buffer with the initial random number pattern.
LIZn	Loads the input buffer with all zeroes.
LLRn	Loads the output buffer with the next random number pattern.
LOn	Loads the output buffer with all ones.
LOIn	Loads the output buffer with the contents of the input buffer.
LZn	Loads the output buffer with all zeroes.

"P" Directives

The P (parameter) directives perform as stated in the following syntax:

$Pp(op)k$

$Pp_1(op)Pp_2$

$PD(op)\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

$PI(op)\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

$PZ=\begin{Bmatrix} k \\ Pp \end{Bmatrix}$

p	specifies one of the 26 parameters (identified A through Z).
(op)	specifies a legal parameter operand.
k	specifies a number (0 through 32,767).

"S" Directive

STPn	Stops program execution and writes a message indicating which Stop directive causes the stop.

"T" Directive

TIME TIME(message)	Writes the system elapsed time and, if applicable, a message to the listed output (LO) file.

6.6 Gould GPDC Synchronous Line Interface Module Diagnostic Unique Directives

The Gould GPDC Synchronous Line Interface Module Diagnostic contains unique directives which perform functions specific to the program.

The Gould GPDC Synchronous Line Interface Module Diagnostic unique directives follow.

6.6.1 "A" Directives

APR

Inserts the SLIM channel address associated with the primary receive channel into the device definition table (DDT) and creates the Command Device (CD) and Test Device (TD) instructions necessary to communicate with the unit under test.

APT

Inserts the SLIM channel address associated with the primary transmit channel into the DDT and creates the CD and TD instructions necessary to communicate with the unit under test.

ASR

Inserts the SLIM channel address associated with the secondary receive channel into the DDT and creates the CD and TD instructions necessary to communicate with the unit under test.

AST

Inserts the SLIM channel address associated with the secondary transmit channel into the DDT and creates the CD and TD instructions necessary to communicate with the unit under test.

6.6.2 "B" Directives

BFI='address'

Relocates the program's input buffer to the specified address. The address may be any address within 16 MB of memory if a Model 9103 GPMC is configured, or anywhere within 512 KB of memory if a Model 9104 GPMC is configured. The length of the relocatable buffer is identical to the default program buffer length.

BFI='PGM'

Relocates the program's input buffer back to the default position within the program.

BFO='address'

Relocates the program's output buffer to the specified address. The address may be any address within 16 MB of memory if a Model 9103 GPMC is configured, or anywhere within 512 KB of memory if a Model 9104 is configured. The length of the relocated buffer is identical to the default program buffer length.

BFO='PGM'

Relocates the program's output buffer back to the default position within the program.

6.6.3 "C" Directives

CCH

Issues a Command Device Start I/O instruction to transmit a Connect Communications Channel order (03_{16}) to the currently assigned SLIM channels. If required, the program performs an implicit Write Control Information function prior to issuing the Connect Communication Channel order and writes the following message to the operator communications (OC) file:

```
'taskname' CONNECT COMMUNICATIONS LINK [C,T] ?
```

Connect either the jumper cable (for jumpered half-duplex assignments) or the modems (for modem half-duplex assignments) before continuing execution. Respond with C to continue program execution or with T to terminate execution.

CCIB='arg_1,...,arg_n'

Modifies the transmit and receive channel control information blocks (CIB) according to the specified arguments 'arg_1,...,arg_n'.

CIB

Writes the contents of the CIB for the assigned SLIM channel to the listed output (LO) file.

6.6.4 "D" Directives

DCB

Writes the contents of the device control block (DCB) for the assigned SLIM channel to the listed output (LO) file. DCB elements are:

Word 0	FLAG:	Flag Word
Word 1	TLIS:	Test Last Interrupt Status (this cell is used only in the DCB for GPDC 0 for each GPMC)
Word 2	CMND:	Last command issued to the GPDC
Word 3	CLQP:	IOCL Queue Pointer
Word 4	DCEP:	DCB Extension (DCE) Pointer
Word 5	TDVS:	Test Device Status
Word 6	TCLA:	Test Command List Address Status
Word 7	TBAS:	Test Buffer Address Status

DCE

Writes the contents of the device control block extension (DCE) for the assigned SLIM channel to the LO file. DCE elements are:

Word 0	UNIT:	Unit word
Word 1	CIBP:	Control Information Block (CIB) pointer
Word 2	TBC:	Transfer Byte Count Accumulator (sum of individual initial byte counts)
Word 3	CLQP:	IOCL Queue Pointer
Word 4	IOTT:	I/O Termination System Elapsed Time
Word 5	TDVS:	Expected Test Device Status
Word 6	TCLA:	Expected Terminal IOCL Address
Word 7	TBAS:	Expected Terminal Buffer Address

DCH

Issues a Command Device (CD) Start I/O instruction to transmit a Disconnect Communications Channel order (07_{16}) to each assigned SLIM channel.

DCIB

Writes the contents of the default control information block (DCIB) to the LO file.

DCIB='arg_1,...,arg_n'

Modifies the contents of the default, transmit, and receive channel control information blocks according to the arguments 'arg_1,...,arg_n'.

DQ

Writes the contents of the I/O Command List (IOCL) queue for the assigned SLIM channel to the listed output (LO) file in the following format:

```
IOCD(ii) :ooffcccc 000bbbbbb
```

ii	specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.
oo	specifies the contents of the order byte in IOCD Word 1.
ff	specifies the contents of the flag byte in IOCD Word 1.
cccc	specifies the hexadecimal value of the initial byte count in IOCD Word 1.
bbbbbb	specifies the hexadecimal value of the initial buffer address in IOCD Word 2.

If no IOCL queue exists, the program writes the following message:

```
IOCL QUEUE EMPTY
```

6.6.5 "H" Directive

HIO

Issues a Halt I/O command to the currently assigned SLIM channel.

6.6.6 "I" Directive

IOR

Issues a Command Device (CD) I/O Reset instruction.

CAUTION

An I/O Reset instruction causes the program to generate a master clear for all GPDC controllers configured on the assigned GPMC. Therefore, executing an I/O Reset in one GPDC program can cause errors to occur in other GPDC programs executing concurrently.

6.6.7 "M" Directive

MODn

Aligns all internal parameters consistent with the model number n as follows:

n	Parameters
9103	16 megabyte GPMC parameters
9104	512 kilobyte GPMC parameters

6.6.8 "R" Directives

RCI

Restores the contents of the default control information block (CIB) to the transmit and receive channel CIBs.

RCIB='arg_1,...,arg_n'

Modifies the contents of the receive channel CIB according to the specified arguments 'arg_1,...,arg_n'.

RDCB

Writes the contents of the device control block for the SLIM channel currently assigned as the receive channel to the listed output (LO) file.

RDCE

Writes the contents of the device control block extension for the SLIM channel currently assigned as the receive channel to the LO file.

6.6.9 "S" Directive

ST

Uses the contents of parameter S to append a subtest designation to the current test number. Parameter S must lie in the range of one through 26. The subtest designation is the letter corresponding to the number in parameter S (e.g., if parameter S is equal to one, the subtest designation is A).

6.6.10 "T" Directives

TBA

Issues a Command Device (CD) Test Buffer Address instruction and writes the returned status to the listed output (LO) file in the following format:

```
TBA STATUS (aa-c-00) :00bbbbbb
```

aa	specifies the two-character hexadecimal GPMC Controller address.
c	specifies the one-character hexadecimal SLIM channel address assigned.
bbbbbb	specifies the current buffer address maintained in the GPMC's auxiliary memory as a result of the last CD Start I/O instruction executed.

TCIB

Writes the contents of the control information block (CIB) for the SLIM channel currently assigned as the transmit channel to the LO file.

TCIB='arg_1,...,arg_n'

Modifies the contents of the transmit channel CIB according to the arguments 'arg_1,...,arg_n'.

TCL

Issues a CD Test Control List Address instruction and writes the returned status to the LO file in the following format:

```
TCL STATUS (aa-c-00):00cccccc
```

aa	specifies the two-character hexadecimal GPMC Controller address assigned.
c	specifies the one-character hexadecimal SLIM channel address.
cccccc	specifies the current IOCL address maintained in the GPMC's auxiliary memory as a result of the last CD Start I/O instruction executed.

TDCB

Writes the contents of the device control block for the SLIM channel currently assigned as the transmit channel to the LO file.

TDCE

Writes the contents of the device control block extension (DCE) for the SLIM channel currently assigned as the transmit channel to the listed output (LO) file.

TDV

Issue a Command Device (CD) Test Device Status instruction and writes the status returned to the LO file in the following format:

```
TDV STATUS (aa-c-00):ddddcccc
```

aa — specifies the two-character hexadecimal GPMC Controller address assigned.

c — specifies the two-character hexadecimal GPMC Controller address assigned.

dddd — specifies the hexadecimal value of the current device-dependent status maintained in the GPMC's auxiliary memory as a result of the last CD Start I/O instruction executed.

TDV device-dependent status definitions (expressed in positional notation) are:

```
dddd = 8... LINK DISCONNECTED
       4... CRC ERROR
       2... FORMAT ERROR
       1... OVERFLOW/UNDERFLOW
       .8.. VRC ERROR
       .4.. RING DETECTED
       .2.. LOSS OF SYNC
       .1.. CHANNEL INOP/NOT INITIALIZED
       ..8. N/A
       ..4. CARRIER OFF
       ..2. TRANSMISSION ERROR
       ..1. INCORRECT LENGTH
       ...8 UNUSUAL END
       ...4 ILLEGAL ORDER
       ...2 INTERRUPT PENDING
       ...1 CHANNEL END
```

cccc — specifies the hexadecimal value of the current remaining byte count as maintained in the GPMC's auxiliary memory as a result of the last CD Start I/O instruction executed.

TIO

Issues a Test Device (TD) Test I/O instruction and writes the returned status (condition code transfer only) to the listed output (LO) file in the following format:

```
TIO STATUS (aa-b-00):c
```

aa — specifies the two-character hexadecimal GPMC Controller address.

b — specifies the one-character hexadecimal SLIM channel address.

c — specifies the hexadecimal value of the condition code returned status.

```
c = 08 UNDEFINED (ALWAYS 0)
    04 GPDC CHANNEL ABSENT
    02 GPDC CHANNEL BUSY
    01 TBA, TCLA, TDV STATUS TRANSFER NOT COMPLETE
```

6.6.11 "W" Directives

Wn

Constructs an I/O Command List (IOCL) write queue and issues a Command Device (CD) Start I/O instruction to initiate a write operation to the currently assigned SLIM channel. Program construction of the IOCL queue is data driven and based on the contents of the transmit channel control information block and the output buffer. If n is specified, the program examines n bytes of the output buffer in the construction of the IOCL queue. If n is not specified, or if parameter W equals zero, the program examines one byte.

WCI

Issues a CD Start I/O instruction to initiate a Write Control Information operation to each currently assigned SLIM channel. The program writes the contents of the transmit channel control information block (CIB) to the primary transmit channel and, if the devices are assigned full-duplex, to the secondary transmit channel. Similarly, it also writes the contents of the receive channel CIB to the primary receive channel and, if full-duplex, to the secondary receive channel.

6.6.12 "X" Directive

XFRn

Constructs I/O Command List (IOCL) read and write queues and issues a Command Device (CD) Start I/O instruction to initiate read and write operations on the currently assigned SLIM channels. In Half-Duplex mode, the program transfers data from the primary transmit channel to the primary receive channel. In Full-Duplex mode, it transfers data concurrently from the primary transmit channel to the secondary receive channel and from the secondary transmit channel to the primary receive channel.

Program construction of the IOCL queues is data driven and based on the contents of the transmit and receive control information blocks (CIB) and the output buffer. The program constructs one IOCL queue for write operations. This queue is also used by the primary transmit channel in Half-Duplex mode and by both the primary and secondary transmit channels in Full-Duplex mode.

For read operations, the program segments the input buffer into three partitions. The program uses the first partition to receive data from the primary receive channel, and the second partition to receive data (in Full-Duplex mode) from the secondary receive channel. The third partition contains the expected input data pattern computed by the program.

The program constructs one IOCL queue to read data from the primary receive channel into the primary input buffer partition. In Full-Duplex mode, the program constructs an additional IOCL queue to read data from the secondary receive channel into the secondary input buffer partition.

If n is specified, the program examines n bytes of the output buffer in the construction of the IOCL queues. If n is not specified, the program examines parameter W bytes or, if parameter W equals zero, it examines one byte.

The algorithm the program uses to compute the expected input data pattern imposes a 512-byte limit on the amount of data actually transferred and read between the devices. If the 512-byte limit is exceeded because of DLE character insertion, the program truncates the transfer count in the transmit and/or receive IOCL queues to limit the data actually transferred or received to 512 bytes.

6.7 Control Information

The Gould Synchronous Line Interface Module (SLIM) Diagnostic contains three control information blocks (CIB): a transmit channel CIB (TCIB), a receive channel CIB (RCIB), and a default CIB (DCIB).

The transmit channel CIB contains control information used to condition the transmit channel.

The receive channel CIB contains control information that conditions the receive channel.

The default CIB contains control information used by the restore control information RCI directive to reset the transmit channel and receive channel CIBs to default (and compatible) values.

The Gould SLIM Diagnostic's default CIB values are:

CIB Byte	Value	Description
MODE (Byte 0)	34_{16}	VRC Parity - Enable (ON) VRC Parity - Even Character Length - 8 bits Strip Sync - Enabled (Strip initial sync character on receive)
NYSN (Byte 1)	01_{16}	Number of SYNC characters - 1
FRMT (Byte 2)		Ring - Disabled (OFF) Send and Test One-Second SYNC - Disabled (OFF) Duplex - Half Line Control - Private Block Check Character - No BCC
DLE (Byte3)	10_{16}	DLE character (CTRL-P)
PAD (Byte 4)	FF_{16}	PAD character
SYNC (Byte 5)	16_{16}	Sync character (CTRL-V)

The program explicitly writes control information to all currently assigned GPDC SLIM channels if it executes a Write Control Information (WCI) directive. The program also implicitly writes control information to a GPDC SLIM channel prior to issuing any Command Device (CD) Start I/O instruction if these conditions occur:

- The program has not previously written control information to the controller.
- There is a change in the contents of the CIB for the SLIM channel since the CIB was last written.
- The Diagnostic Executive Program (DEXP) is reinitialized by entering RST and RUN on the operator's console.

6.8 Control Information Block (CIB) Directives

Use the following directives to modify the contents of the transmit, receive, and default CIBs:

TCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the channels transmit CIB.

RCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the channels receive CIB.

CCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of both the transmit and receive channels CIBs.

DCIB='arg_1,arg_2,...,arg_n'

Modifies the contents of the default, transmit, and receive channels CIBs.

The directive arguments (arg_1,arg_2,...,arg_n) specify modifications made to the CIBs. The notation (parameter) specifies an optional parameter within an argument.

6.9 Control Information Block (CIB) Directive Arguments

Use the following arguments with CIB directives:

NOBCC

No Block Check Character. The block check character field of the specified CIB is set to the value 07_{16}.

CRC16

Select CRC-16 Block Check Character. The block check character field of the specified CIB is set to the value 00_{16}.

LRC8

Select LRC-8 Block Check Character. The block check character field of the specified CIB is set to the value 05_{16}.

CRCCCIT

Select CRC-CCITT Block Check Character. The block check character field of the specified CIB is set to the value 06_{16}.

CLP(cl,p)

Sets the character length/parity (CLP) field of the specified CIB to select the character length specified by the value of cl and the parity specified by the optional parameter p as follows:

cl	p	CLP Field
6	No Parity	00_{16}
6	Even	00_{16}
6	Odd	01_{16}
7	No Parity	02_{16}
8	No Parity	03_{16}
7	Even	04_{16}
7	Odd	05_{16}
8	Even	06_{16}
8	Odd	07_{16}

DLE=xx

Sets the DLE character in the specified CIB to the hexadecimal value xx.

PAD=xx

Sets the PAD character in the specified CIB to the hexadecimal value xx.

SYN(s-0,s-1,s-2)

Sets the SYN0 character in the specified CIB to the hexadecimal value s-0, the SYN1 character to the value s-1 (if specified), the SYN2 character to the value s-2 (if specified), and the NSYN byte to the total number of sync characters specified.

RN/ON

Sets the ring detection enable (RN) bit of the specified CIB to a one (Ring Detection Enabled).

RN/OFF

Sets the ring detection enable (RN) bit of the specified CIB to a zero (Ring Detection Disabled).

RTS/ON

Sets the inhibit RTS delay bit of the specified CIB to a one (RTS Delay Inhibited).

RTS/OFF

Sets the inhibit RTS delay bit of the specified CIB to a zero (RTS Delay Enabled).

S/ON

Sets the switched line (S) bit of the selected CIB to a one (Switched Line).

S/OFF

Sets the switched line (S) bit of the selected CIB to a zero (Private Line).

SN/ON

Sets the send and test one second sync (SN) bit of the specified CIB to a one (ON).

SN/OFF

Sets the send and test one second sync (SN) bit of the specified CIB to a zero (OFF).

SS/ON

Sets the strip initial sync character (SS) bit of the specified CIB to a one (ON).

SS/OFF

Sets the strip initial sync character (SS) bit of the specified CIB to a zero (OFF).

RC(rc_1/rc_2/.../rc_n)

Appends the recognition characters rc_1 through rc_n to the specified CIB.

Each recognition character field is in the form:

char:cff_1,...,cff_n

char specifies the character to be recognized. The character is any valid ASCII character or control character (except a single quotation mark preceded by a double quotation mark). Specify an ASCII control character by preceding the character with a double quotation mark which causes the character to perform a logical AND operation with the value $3F_{16}$.

cff specifies the condition/function triplet field in the following form:

cond,$func_0$,$func_1$

cond specifies a two-digit hexadecimal number representing the conditions under which the character (char) is to be recognized.

$func_0$ specifies a two-digit hexadecimal number representing the functions in function byte 0 of the CIB.

$func_1$ specifies a two-digit hexadecimal number representing the functions in function byte 1 of the CIB.

BISYNC

Loads the specified CIBs with the control information necessary to implement binary synchronous communications (BISYNC) protocol.

Note: No other arguments can be specified with BISYNC.

The control information required to implement binary synchronous (BISYNC) protocol follows:

CONTROL INFORMATION FOR BINARY SYNCHRONOUS COMMUNICATIONS PROTOCOL

```
MODE:34      NSYN:02  FRMT:48      DLE:10     PAD:FF
SYN0:16      SYN1:16
CHAR:01      COND:60  FNC0:90      FNC1:20
CHAR:02      COND:E7  FNC0:D0      FNC1:20
     COND:60 FNC0:90  FNC1:20
CHAR:03      COND:D7  FNC0:3C      FNC1:10
     COND:60 FNC0:1C  FNC1:00
CHAR:04      COND:40  FNC0:00      FNC1:81
CHAR:05      COND:D7  FNC0:34      FNC1:10
     COND:60 FNC0:04  FNC1:00
CHAR:10      COND:D7  FNC0:10      FNC1:00
     COND:50 FNC0:00  FNC1:AF
     COND:30 FNC0:00  FNC1:A7
     COND:60 FNC0:00  FNC1:A7
CHAR:15      COND:60  FNC0:24      FNC1:00
CHAR:16      COND:D7  FNC0:12      FNC1:28
     COND:C2 FNC0:00  FNC1:8B
     COND:C3 FNC0:10  FNC1:08
     COND:60 FNC0:02  FNC1:28
CHAR:17      COND:D7  FNC0:3C      FNC1:10
     COND:60 FNC0:1C  FNC1:00
CHAR:1F      COND:60  FNC0:18      FNC1:00
CHAR:3B      COND:E7  FNC0:24      FNC1:00
     COND:60 FNC0:10  FNC1:00
CHAR:30      COND:E7  FNC0:24      FNC1:00
     COND:60 FNC0:10  FNC1:00
CHAR:31      COND:E7  FNC0:24      FNC1:00
     COND:60 FNC0:10  FNC1:00
CHAR:3C      COND:E7  FNC0:24      FNC1:00
     COND:60 FNC0:10  FNC1:00
CHAR:FF      COND:C1  FNC0:00      FNC1:AA
     COND:C2 FNC0:04  FNC1:08
```

CHAPTER 7

I/O COMMAND TRACE AND DATA RECORDER

7.1 General Overview

The Gould GPDC Synchronous Line Interface Module Diagnostic contains optional capabilities that provide comprehensive listings of I/O activity during program execution.

These capabilities include I/O Command Trace and Data Recorder.

When the appropriate options are set, these routines produce a visual representation to the listed output (LO) file of the data being manipulated and I/O commands being issued.

Use these listings to examine, in greater detail, the processor activity by supplementing error messages and verifying that the actual data being generated coincides with the commands being issued.

7.2 I/O Command Trace

The I/O Command Trace provides a record of all I/O instructions issued and in what order they are issued during program execution. Option 17 enables the I/O Command Trace and sends the information to the LO file.

7.2.1 Starting Execution

To begin I/O Command Trace execution, set Option 17 (enables Command Trace) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the LO file:

```
<<<<'taskname'----hh:mm:ss>>>>
TST:ttt,DEV:aa-c-00,PC:pppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal GPMC Controller address.
c	specifies the one-character hexadecimal SLIM channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.2.2 Milestone Messages

The following milestone messages are written to the listed output (LO) file during an I/O Command Trace:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx	specifies the instruction issued.
qqqqqq	specifies the address of the IOCL queue.

The IOCL queue contents (repeated for all IOCDs within the IOCL queue) are written:

```
IOCD(ii):       ooffcccc        ff00cccc
```

ii	specifies the queue slot number of the I/O command doubleword (IOCD) in the IOCL queue.
oo	specifies the contents of the order byte in IOCD Word 1.
bbbbb	specifies the initial buffer address in IOCD Word 1.
ff	specifies the contents of the flag byte in IOCD Word 2.
cccc	specifies the hexadecimal value of the initial byte count in IOCD Word 2.

7.2.3 Terminating the I/O Command Trace

To terminate the I/O Command Trace, issue an Attention command and reset Option 17 (enables Command Trace) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 17
```

7.3 Data Recorder

The Data Recorder provides a dump of the actual data being manipulated during program execution. The dump shows how the data appears during I/O activity. Option 18 enables Data Recorder. With this option set, the routine sends a dump of data to the listed output (LO) file.

7.3.1 Starting Execution

To begin Data Recorder execution, set Option 18 (enables Data Recorder) using the DEXP Set Option command. During execution, the program writes the following task separation banner and header to the listed output (LO) file:

```
<<<<'taskname'----hh:mm:ss>>>>
TST:ttt, DEV:aa-c-00, PC:ppppp
DIRTV:directive
```

ttt	specifies the test number being executed.
aa	specifies the two-character hexadecimal controller address.
c	specifies the one-character hexadecimal channel address.
ppppp	specifies the program counter value within the program.
directive	specifies the directive causing the I/O operation (for directive tests only).

7.3.2 Milestone Messages

The following milestone messages are written during the Data Recorder routine:

The I/O command issued and the I/O Command List (IOCL) queue pointer (if applicable):

```
CMND:xxxxxxxx,IOCLQP:00qqqqqq
```

xxxxxxxx	specifies the instruction issued.
qqqqqq	specifies the address of the IOCL queue.

The I/O command doubleword (IOCD) specification (for each IOCD in the IOCL queue):

```
IOCD  (ADDR)  (WRD1)     (WRD2)    ORDER IS:   order
(ii)  qqqqqq  ooffcccc   00bbbbbb  FLAGS ARE:  flag
                                       .
                                       .
                                       .
                                     flag
```

ii	specifies the queue slot number of the IOCD in the IOCL queue.
qqqqqq	specifies the address of the IOCD in the IOCL queue.
oo	specifies the contents of the order byte in IOCD Word 1.
bbbbbb	specifies the hexadecimal initial buffer address in IOCD Word 1.
ff	specifies the contents of the flag byte in IOCD Word 1.

cccc — specifies the hexadecimal value of the initial byte count in IOCD Word 2.

order — specifies the IOCD order. If the listed output (LO) file assignment is to a teletypewriter, the program writes the hexadecimal value of the order byte and a text description of the order as follows:

oo = 00	STOP
01	TRANSMIT DATA
02	RECEIVE DATA
03	CONNECT COMMUNICATIONS CHANNEL
05	WRITE CONTROL INFORMATION
07	DISCONNECT COMMUNICATIONS CHANNEL
0F	RING OUT
80	TRANSFER-IN-CHANNEL
Other	ILLEGAL

flag — specifies the IOCD flags. If the listed output (LO) file assignment is to a teletypewriter, the program writes the hexadecimal value of the flag byte. Otherwise, a text description of each flag bit set is written as follows:

ff = 40	INTERRUPT ON ZERO BYTE COUNT
20	HALT ON TRANSMISSION ERROR
10	SUPPRESS INCORRECT LENGTH
08	INTERRUPT ON UNUSUAL END
04	COMMAND CHAIN
02	DATA CHAIN
01	INTERRUPT ON CHANNEL END

The message for each I/O command doubleword (IOCD) involving a data transfer:

```
IOCD  (ADDR)   DATA LENGTH SPECIFIFES:cccccTRANSFERRED:tttt
(ii) bbbbbb    mmm mm a a a . . a a mmm mmm
       (Rnnnnn) xx..xx..xxxxxx....xxxxxx..xx.
```

ccccc	specifies the decimal value of initial byte count in IOCD Word 1.
ttttt	specifies the number of decimal bytes actually transferred (only written for the last executed IOCD in the I/O Command List (IOCL) queue).
ii	specifies the queue slot number of the IOCD in the IOCL queue.
bbbbbb	specifies the hexadecimal initial buffer address in IOCD Word 2.
Rnnnnn	specifies the relative frame number of the first data byte in the line.
mmm	specifies the mnemonic name of an ASCII control character associated with the hexadecimal representation of the ASCII data byte ..xx.. .
a	specifies the graphic character associated with the hexadecimal representation of an ASCII data byte ..xx.. .
..xx..	specifies the hexadecimal representation of an ASCII data byte.

The message for the terminal IOCD:

```
IOCD  (ADDR)  DEVICE STATUS AT TERMINAL IOCL: status
(ii)  qqqqqq                                    .
                                                .
                                                .
                                              status
```

ii	specifies the queue slot number of the last executed IOCD in the IOCL queue.
qqqqqq	specifies the address of the last executed IOCD in the IOCL queue.
status	specifies the test device (TDV) status at I/O completion. If the value of the TDV status is 0000_{16}, the program writes one of the following messages:

I/O COMPLETE Normal completion.

I/O TIME-OUT I/O not completed within expected time, HALT I/O issued to terminate operation.

If the value of the test device (TDV) status is other than 0000_{16} and the listed output (LO) file assignment is to a teletypewriter, the program writes the hexadecimal value of the TDV status. Otherwise, the program writes a text description of each TDV status bit set as follows:

```
8...    LINK DISCONNECTED
4...    CRC ERROR
2...    FORMAT ERROR
1...    OVERFLOW/UNDERFLOW
.8..    VRC ERROR
.4..    RING DETECTED
.2..    LOSS OF SYNC
.1..    CHANNEL INOP/NOT INITIALIZED
..8.    N/A
..4.    CARRIER OFF
..2.    TRANSMISSION ERROR
..1.    INCORRECT LENGTH
...8    UNUSUAL END
...4    ILLEGAL ORDER
...2    INTERRUPT PENDING
...1    CHANNEL END
```

7.3.3 Terminating Data Recorder

To terminate the Data Recorder routine, issue an Attention command and reset Option 18 (enables Data Recorder) using the DEXP Reset command.

Example

```
@@A
DXP>RESET taskname,OPT 18
```

Gould Vector Processor

Common Memory Interface

(RM.CML.r)

Diagnostic Description

Diagnostic Number: 111032-286r

Documentation Release Level: 1.0

Documentation Release Date: May 1985

Publication Order Number: 326-005730-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1985
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould Vector Processor Common Memory Interface (RM.CMI.r) Diagnostic Description Release 1.0, Publication Order Number **326-005730-000**, was printed May 1985.

The manual contains the following pages:

Title page
Copyright page
iii/iv through v/vi
1-1/1-2
2-1 through 2-5/2-6
3-1/3-2
4-1 through 4-2
5-1 through 5-4

CONTENTS

CHAPTER 1 PROGRAM OVERVIEW

1.1 General Overview 1-1
1.2 Resource Requirements 1-1

CHAPTER 2 PREPARING FOR EXECUTION

2.1 Assignments 2-1
2.2 Program Initialization 2-1
2.3 Program Option Set 2-4

CHAPTER 3 PROGRAM EXECUTION

3.1 Starting Program Execution 3-1
3.2 Milestone Messages 3-1
3.3 Terminating Program Execution 3-1

CHAPTER 4 ERROR REPORTING

4.1 Standard Error Messages 4-1
4.2 Unique Error Messages 4-2

CHAPTER 5 TEST DESCRIPTIONS

5.1 Summary of Tests 5-1
5.2 Test 1 Address in Address 5-1
5.3 Test 2 Address Complement in Address 5-1
5.4 Test 3 All Zeros 5-1
5.5 Test 4 All Ones 5-1
5.6 Test 5 Alternate Zeros 5-1
5.7 Test 6 Alternate Ones 5-1
5.8 Test 7 Walking Zero Test 5-2
5.9 Test 8 Walking One Test 5-2
5.10 Test 9 Byte Access 5-2
5.11 Test 10 Halfword Access 5-2
5.12 Test 11 Doubleword Access 5-2
5.13 Test 12 Pseudo Register X'7E28' (Address Translate) 5-3
5.14 Test 13 Pseudo Register X'7E08' (Interrupt) 5-3
5.15 Test 14 Write/Read Using Operator Parameters 5-4

CHAPTER 1

PROGRAM OVERVIEW

1.1 General Overview

The Gould Vector Processor Common Memory Interface Diagnostic (RM.CMI.r) is a level two macrodiagnostic program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This macrodiagnostic program contains a sequence of 14 tests which verify the correct operation of the vector processor common memory interface. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The program requires a minimum of operator intervention. Skip tests requiring intervention by setting appropriate options.

During execution, the program writes milestone messages and error message sequences to the listed output (LO) file which the program assigns by default to the operator's console.

1.2 Resource Requirements

The Gould Vector Processor Common Memory Interface Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory
- A Vector Processor (Model 3300)

This program is invalid for CONCEPT 32/67 and Virtual Memory (V6) processors.

CHAPTER 2

PREPARING FOR EXECUTION

2.1 Assignments

By entering a Return key in response to the prompts, the Gould Vector Processor Common Memory Interface Diagnostic assigns the following default values during initialization:

- VP IPCA base address is 00 (and must be assigned)
- Enable interrupt is set to NO
- Interrupt level is 00

The program computes the first and last CMI addresses and enters them as default values in the initialization sequence.

To modify the default values, respond to the initialization prompts with one of the responses written with the prompt or use the DEXP Assign command.

Note: Values modified during initialization or by an Assign command become the default values for all reinitialization sequences until reloading and initializing the program.

2.2 Program Initialization

Before initializing the Gould Vector Processor Common Memory Interface Diagnostic, load it into the system by following the loading procedures in the appropriate Diagnostic Facility User's Manual. The program name for loading the Gould Vector Processor Common Memory Interface Diagnostic is RM.CMI.r. After the program loads, it writes the following initialization sequence to the operator communications (OC) file:

Note: The program reissues any incorrectly answered initialization sequence prompts.

1. The program writes the following message:

   ```
   'taskname' INITIALIZATION
   ```

   The program then determines (from the CPU status word) the CPU type under which it is loaded, and writes the following message identifying the CPU type:

   ```
   CPU TYPE = 32/xx
   ```

   xx specifies the machine type as determined by the program.

   If the CPU type identified is a 32/67 or a Virtual Memory (V6) processor, the program writes the following message and terminates execution:

   ```
   'taskname' FATAL ERROR - THIS DIAGNOSTIC IS NOT
              APPLICABLE TO THIS MACHINE TYPE.
   ```

2. If a valid CPU type is identified, the program continues the initialization sequence by issuing the following prompt:

```
VP IPCA BASE ADDRESS=bbbbbb ; [CR,?,######]?
```

Responding with the Return key causes the program to assign the specified default address (bbbbbb) which is either zero (following the load sequence) or equal to the last value specified during a Reinitialization sequence. To modify the default base address, enter the physical SelBUS address corresponding to the base address of the Vector Processor's interprocessor communications area (IPCA).

Ensure that the IPCA address meets the following conditions:

- No overlay of the DEXP or the program occurs.
- If performing an address-in-address write/read sequence and referring to the IPCA, the data read is zero.
- If the SelBUS address is written to memory location $7E28_{16}$ biased by the specified address and read from the same memory location, a SelBUS-to-Vector Processor address translation occurs.

Responding with a question mark (?) to this prompt and subsequent prompts causes the program to write a description of valid prompt responses then to reissue the prompt.

3. The program then writes the following Common Memory Interface (CMI) profile to the listed output (LO) file:

```
  VECTOR PROCESSOR CMI PROFILE
 SELBUS ADDR          VP BUS ADDR
================================
ffffff-eeeeee IPCA b-aaaaa-ccccc
```

ffffff	specifies the first SelBUS address on the designated Vector Processor bus.
eeeeee	specifies the last SelBUS address on the designated Vector Processor bus.
IPCA	specifies the interprocessor communication area (IPCA).
b	specifies the Vector Processor bus number.
aaaaa	specifies the Vector Processor halfword address corresponding to the SelBUS address ffffff.
ccccc	specifies the Vector Processor halfword address corresponding to the SelBUS address eeeeee.

The program develops the profile using the pseudo register $7E28_{16}$ translate address function. The values specified reflect the actual capacity of the Vector Processor CMI as seen by the Vector Processor.

4. The program then issues the following prompt:

```
FIRST CMI ADDRESS=ffffff;[CR,?,######]?
```

Responding with the Return key causes the program to assign the specified default first CMI address (ffffff) computed by the program to be either 8192W above the IPCA base address (the defined length of the IPCA) following the load sequence or equal to the last value specified during a Reinitialization sequence. To modify the default first CMI address, enter a valid address.

Ensure that the address meets the following conditions:

- No overlay of the DEXP, the program, or the interprocessor communications area (IPCA) occurs.
- If performing an address-in-address write/read sequence, the data read equals the data written (i.e., RAM).
- If the SelBUS address is written to memory location $7E28_{16}$ biased by the IPCA base address and read from the same location, a SelBUS-to-Vector Processor address translation occurs but, an address out of VP range indication fails to occur during the translation.

5. The program then issues the following prompt:

```
LAST CMI ADDRESS=eeeeee;[CR,?,######]?
```

Responding with the Return key causes the program to assign the specified default address (eeeeee) computed by the program using the Vector Processor address translate function to be either the last Common Memory Interface (CMI) address recognized by the Vector Processor following the load sequence or equal to the last value specified during a Reinitialize command sequence. To modify the default last CMI address, enter a valid address.

Ensure that the address meets the following conditions:

- No overlay of the DEXP, the program, or the IPCA occurs, and the address is not less than the first CMI address to test. The address may be equal to the first CMI address to test because the program executes on a single address range.
- If performing an address-in-address write/read sequence, the data read equals the data written (i.e., RAM).

6. The program then issues the following prompt:

   ENABLE INTERRUPT=NO;[CR,Y,N]?

   Responding with the Return key causes the program to assign the specified default Enable interrupt (no). To modify the default Enable interrupt, enter Y (yes). Specifying Y (yes) implies that an external interrupt cable is connected between the Vector Processor and the Real-time Option Module (RTOM) within the CPU. Specifying N (no) implies the cable connection is not made or the external interrupt test associated with pseudo register $7E08_{16}$ in the interprocessor communications area (IPCA) is not to be run.

   If Y (yes) is specified, the program issues the following prompt:

   INTERRUPT LEVEL=00;[CR,?,##]?

   Responding with the Return key causes the program to assign the specified default interrupt level (00). To modify the default interrupt level, enter the valid two-character hexadecimal interrupt level (2A through 7F).

7. The program then writes the following message:

   'taskname' INITIALIZATION COMPLETE

   The message indicates that initialization is successful. At this time, modify options and/or start the program.

2.3 Program Option Set

Once loaded and initialized, the Gould Vector Processor Common Memory Interface Diagnostic enters a wait state.

Use the DEXP Set and Reset commands to change default option settings, thereby modifying execution conditions.

The Gould VP Common Memory Interface Diagnostic contains the following options:

Option	Default	Effect If Set
0	Reset	Inhibits error messages. Setting this option inhibits all error messages except the functions of Options 1 through 4 or a message in progress.
1	Reset	Rings bell on error. Setting this option rings the bell on the operator's console once per error.
2	Reset	Stops on error. Setting this option suspends execution if the program detects an error. Prior to suspension, the program writes an associated error message and issues the following prompt: 'taskname' ERROR..., [C,H,T]? Respond with C to continue execution, with H to continue suspension, or with T to terminate execution.

Option	Default	Effect If Set
3	Reset	Loops on error. Setting this option causes the program to repeat the sequence which detects an error until obtaining a correct result. During the repeat function, no operand values change. If Options 0, 1, and 2 are set, the program performs their functions prior to this function.
4	Reset	Loops on subtest. Setting this option repeats the current subtest until resetting the option. During the course of the repeat function, no operand values change.
5	Reset	Loops on test. Setting this option repeats the current test until resetting the option.
6	Set	Repeats program. Setting this option repeats the entire program until resetting the option or until reaching the pass count specified in the DEXP Start/Restart command.
7	Set	Unimplemented.
8	Reset	Inhibits operator intervention tests. Setting this option inhibits any test requiring operator intervention.
9	Reset	Unimplemented.
10	Set	Writes pass count messages. Setting this option writes the program's pass count message to the listed output (LO) file after completing each program pass.
11	Reset	Writes milestone messages. Setting this option writes the program's milestone message to the listed output (LO) file prior to executing each test.
12	Reset	Unimplemented.
13	Reset	Unimplemented.
14	Reset	Unimplemented.
15	Reset	Unimplemented.
16	Reset	Writes all detected data errors.

CHAPTER 3

PROGRAM EXECUTION

3.1 Starting Program Execution

After setting options, issue a Diagnostic Executive program (DEXP) Start command to begin execution. As the program begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#111032-286r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the program begins testing and writes milestone messages to the listed output (LO) file.

3.2 Milestone Messages

If Option 11 (writes milestone messages) is set, the program writes the following message to the LO file prior to the execution of each test:

```
hh:mm:ss 'taskname' TEST:n  (test description)
```

If Option 10 (writes pass count messages) is set, the program writes the following message to the LO file after completing all designated tests:

```
hh:mm:ss 'taskname' FINISHED PASS:n
```

3.3 Terminating Program Execution

Terminate program execution using one of the following methods:

DEXP Abort command

```
@@A
DXP>ABORT taskname
```

DEXP terminates the program. To begin execution again, use the DEXP Start or Restart commands.

DEXP Hold command

```
@@A
DXP>HOLD taskname
```

DEXP places the program on hold. To resume execution at this point, use the DEXP Continue command. To begin execution again, use the DEXP Start or Restart commands.

Note: Using these commands with only one program running on the system causes DEXP to execute the commands and issue a DXP> prompt.

CHAPTER 4

ERROR REPORTING

4.1 Standard Error Messages

The Gould VP Common Memory Interface Diagnostic writes all error messages to the listed output (LO) file, which the program assigns by default to the operator's console. To assign the LO file to a device other than the operator's console, use the DEXP Assign command.

A standard error message routine contains the following components:

1. The message separation banner

```
<<<<'taskname'----hh:mm:ss>>>>
```

   taskname — specifies the one to eight character program name indicated in the load map.

   hh:mm:ss — specifies the system elapsed execution time at the point of error.

2. The error message header

```
TST:ttt, PASS:n,          PC:aaaaa
```

   ttt — specifies the test number.

   n — specifies the current pass count.

   aaaaa — specifies the program counter at the point of error.

4.2 Unique Error Messages

If no interrupt is received during Test 13, the program writes the following message:

```
NO SERVICE INTERRUPT (SI)
```

If the data written does not compare with the data read during Test 14, the program writes the following message:

```
DATA ERROR(S)
ssssss(b-aaaaa) WRITTEN:xxx....xxx READ:yyy....yyy
```

ssssss	specifies the SelBUS address.
b	specifies the Vector Processor (VP) bus number.
aaaaaa	specifies the Vector Processor halfword address.
xxx...xxx	specifies the data pattern written (byte, halfword, word, or doubleword).
yyy...yyy	specifies the data pattern read (byte, halfword, word, or doubleword).

If Option 16 (writes all detected errors) is set, the program writes all data errors detected. Otherwise, the program only writes the first five data errors.

CHAPTER 5

TEST DESCRIPTIONS

5.1 Summary of Tests

The Gould VP Common Memory Interface Diagnostic contains a sequence of 14 tests which run either in a continuous sequence or individually by operator request. The test sequences verify the ability of both the CPU and the Vector Processor to communicate through the Common Memory Interface.

5.2 Test 1 Address in Address

This test verifies the unique addressability of each Common Memory Interface (CMI) location.

5.3 Test 2 Address Complement in Address

This test verifies that the program writes the complement of the SelBUS address into each memory location throughout the range of the interface and checks each location for correct data.

5.4 Test 3 All Zeros

This test verifies that the program writes 00000000_{16} into each memory location throughout the range of the interface and checks each location for correct data.

5.5 Test 4 All Ones

This test verifies that the program writes $FFFFFFFF_{16}$ into each memory location throughout the range of the interface and checks each location for correct data.

5.6 Test 5 Alternate Zeros

This test verifies that data bits are not shorted together. The program writes 55555555_{16} into each memory location throughout the range of the interface and checks each location for correct data.

5.7 Test 6 Alternate Ones

This test verifies that the program writes $AAAAAAAA_{16}$ into each memory location throughout the range of the interface and checks each location for correct data.

5.8 Test 7 Walking Zero Test

This test verifies noise immunity and cross talk within the CMI. The program writes a shifting data pattern (beginning with $7FFFFFFF_{16}$ and continuing with $BFFFFFFF_{16}$ until $FFFFFFFE_{16}$ is reached) into each memory location throughout the range of the interface and checks each location for correct data.

5.9 Test 8 Walking One

This test verifies that the program writes a shifting data pattern (beginning with 80000000_{16} and continuing with 40000000_{16} until 00000001_{16} is reached) into each memory location throughout the range of the interface and checks each location for correct data.

5.10 Test 9 Byte Access

This test verifies the CMI's ability to access data bytes, halfwords, and doublewords by varying the F and C bits. The program writes a binary progression pattern (a word at a time) into each memory location throughout the range of the interface and reads each location one byte at a time checking for correct data.

5.11 Test 10 Halfword Access

This test verifies that the program writes a binary progression pattern (a word at a time) into each memory location throughout the range of the interface and reads the data a halfword at a time checking for correct data.

5.12 Test 11 Doubleword Access

This test verifies that the program writes a binary progression pattern (a word at a time) into each memory location throughout the range of the interface and reads the data a doubleword at a time checking for correct data.

5.13 Test 12 Pseudo Register X'7E28' (Address Translate)

This test verifies the address translation of a Gould SelBUS address into the corresponding Vector Processor bus number and address.

The program issues the following prompt to the operator communications (OC) file:

```
SELBUS ADDR TO TRANSLATE ;[CR,######]?
```

Respond with any Gould SelBUS address. The program then writes the following message:

```
SELBUS ADDR:ssssss = VP BUS:b ADDR:aaaaa
```

sssss specifies the SelBUS address.

b specifies the Vector Processor bus number.

aaaaa specifies the Vector Processor halfword address.

If a SelBUS address is specified that is out of the VP Common Memory Interface (CMI) range, the program writes the following message and reissues the prompt:

```
SELBUS ADDR:ssssss IS NOT WITHIN VP RANGE
```

This sequence can be repeated any number of times.

To exit the test, respond to the prompt by pressing the Return key.

If Option 8 (inhibits operator intervention tests) is set, the program skips this test.

5.14 Test 13 Pseudo Register X'7E08' (Interrupt)

This test causes an interrupt by writing to the pseudo register ($7E08_{16}$) in the VP interprocessor communications area (IPCA).

If no interrupt is received within one-tenth of a second from the time the pseudo register is written, the program writes the following message:

```
NO SERVICE INTERRUPT (SI)
```

If the Enable Interrupt prompt is answered N (no) during the initialization sequence, the program skips this test.

5.15 Test 14 Write/Read Using Operator Parameters

This test allows specification of a SelBUS address and data pattern. The program issues the following prompt to the OC file:

```
SELBUS ADDR TO WRITE/READ ;[CR,#####]?
```

Respond with a valid SelBUS address. If the address is not within the VP Common Memory Interface (CMI) range, the program reissues the prompt.

The program then issues the following prompt:

```
DATA PATTERN ;[CR,########]?
```

Respond with a one to eight hexadecimal digit data pattern. The program writes the data pattern into the memory location specified by the SelBUS address, reads the data pattern, and checks for the correct result. If the data does not compare, the program writes an error message.

This sequence can be repeated any number of times.

To exit the test, respond to the prompt by pressing the Return key.

If Option 8 (inhibits operator intervention tests) is set, the program skips this test.

Gould RPU Disc Processor

(RP.DPD.r)

Peripheral Diagnostic Description

Diagnostic Number: 001032-953r

Documentation Release Level: 1.1

Documentation Release Date: November 1984

Publication Order Number: 326-005310-100

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1984
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould RPU Disc Processor Diagnostic (RP.DPD.r) Peripheral Diagnostic Description, Publication Order Number **326-005310-100,** was printed November 1984.

This manual contains the following pages:

Title page
Copyright page
History page
iii through v/vi
1-1 through 1-2
2-1 through 2-10
3-1 through 3-28
4-1 through 4-17/4-18
5-1 through 5-42
6-1/6-2

CONTENTS

Chapter Page

1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-2

2 ASSIGNMENTS AND INITIALIZATION

2.1 Disc Processor Controller Model Assignment 2-1
2.2 System Disc Flag Assignment 2-1
2.3 Formatted Drive Assignment 2-1
2.4 Unformatted Drive Assignment 2-2
2.5 Initial Drive Checkout on Unformatted Media 2-2
2.6 Media Formatting 2-3
2.7 Single Port Assignment 2-3
2.8 Dual Ported Operating Procedures 2-3
2.9 Dual Ported Mode Test Restrictions 2-5
2.10 Diagnostic Initialization 2-6
2.11 Diagnostic Option Set 2-9
2.12 Diagnostic Execution 2-10

3 TEST DESCRIPTIONS

3.1 Summary of Tests 3-1
3.2 Test 1--Interrupt Instruction Verification 3-1
3.3 Test 2--Service Interrupt Halt I/O, Stop I/O, Test I/O Verification 3-2
3.4 Test 3--Quiescent Device Status Verification 3-2
3.5 Test 4--Label Verification 3-2
3.5.1 Label Verification of the Diagnostic Cylinders 3-2
3.5.2 Label Verification of the Entire Media 3-3
3.6 Test 5--Basic Disc Functions 3-3
3.7 Test 6--Invalid Memory Access 3-3
3.8 Test 7--Channel Program Violation 3-4
3.9 Test 8--Channel Program Violation, Undefined Commands 3-5
3.10 Test 9--Track and Sector Label Flag Verification 3-5
3.11 Test 10--Unsupported Transaction Verification 3-5
3.12 Test 11--Halt I/O, Stop I/O Program Termination 3-6
3.13 Test 12--Variable Length Seeks 3-6
3.14 Test 13--Write/Read Diagnostic Track and Highest Testable Track 3-6
3.15 Test 14--Write/Read Zeros, Ones, Floating Zero Data 3-7
3.16 Test 15--Write/Read Floating Ones, Binary Progression and Regression 3-8
3.17 Test 16--Write/Read Random and ASCII Data 3-9

Chapter Page

3.18 Test 17--Write/Read All Sectors on One Track 3-10
3.19 Test 18--Head Addressing 3-11
3.20 Test 19--Write Disc One Track Per Start I/O 3-11
3.21 Test 20--Read Disc One Track Per Start I/O 3-12
3.22 Test 21--Automatic Head Advance 3-13
3.23 Test 22--Automatic Head Advance, Data Chained Read 3-14
3.24 Test 23--Automatic Track Advance 3-15
3.25 Test 24--Automatic Track Advance, Data Chained Read 3-16
3.26 Test 25--Mode Register Verification 3-17
3.27 Test 26--Random Access Write/Read or Read Labels 3-17
3.28 Test 27--Write Diagnostic Track Labels 3-19
3.29 Test 28--Read Diagnostic Track Labels 3-20
3.30 Test 29--Write Data Chained--Four Sectors/Start I/O 3-20
3.31 Test 30--Read Data Chained--Four Sectors/Start I/O 3-21
3.32 Test 31--Variable Byte Count Zero Fill 3-22
3.33 Test 32--Variable Byte Count Two Sector Read 3-23
3.34 Test 33--Read Variable Byte Count Data Chained 3-23
3.35 Test 34--Write Variable Byte Count Data Chained 3-24
3.36 Test 35--Read, Decrementing Byte Count, Data Chained 3-25
3.37 Test 36--Read, Decrementing Byte Count, Skip Data Chain 3-25
3.38 Test 37--Error Correction Code 3-26
3.39 Test 38--Write Single Sector 3-26
3.40 Test 39--Read Single Sector 3-27
3.41 Test 40--Reserve/Release Drive Verification 3-28

4 ERROR REPORTING

4.1 Error Message Assignment 4-1
4.2 Error Messages and Message Sequence 4-1
4.3 Status Error Report Exceptions 4-4
4.4 Condition Code Errors 4-5
4.5 No Interrupt and Device End Messages 4-6
4.6 Status Address Word Error 4-6
4.7 Program Start Errors 4-7
4.8 Additional Messages 4-8
4.9 Directive Error Reporting 4-9
4.10 Label Verification Messages 4-12
4.10.1 Description of Label Flag Byte One 4-13
4.10.2 Description of Label Flag Byte Two 4-13
4.11 Track Deallocation Messages 4-13

5 DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Language Compiler 5-1
5.2 DEXP Standard Directives 5-1
5.2.1 "A" Directives 5-1
5.2.2 "C" Directives 5-2
5.2.3 "D" Directives 5-2
5.2.4 "H" Directives 5-3
5.2.5 "J" Directives 5-3
5.2.6 "L" Directives 5-4

Chapter Page

5.2.7 "P" Directives 5-6
5.2.8 "S" Directive 5-7
5.2.9 "T" Directive 5-7
5.3 Gould RPU Disc Processer Unique Directives 5-8
5.3.1 "A" Directives 5-8
5.3.2 "B" Directives 5-9
5.3.3 "C" Directives 5-9
5.3.4 "D" Directives 5-12
5.3.5 "F" Directives 5-15
5.3.6 "H" Directives 5-16
5.3.7 "I" Directives 5-16
5.3.8 "L" Directives 5-16
5.3.9 "M" Directive 5-18
5.3.10 "N" Directive 5-19
5.3.11 "R" Directives 5-19
5.3.12 "S" Directives 5-21
5.3.13 "T" Directive 5-24
5.3.14 "W" Directives 5-27
5.3.15 "Z" Directives 5-28
5.4 Directives Issuing Start I/O Instructions 5-29
5.5 Directive Arguments 5-29
5.5.1 IOCD Flag Byte Arguments 5-29
5.5.2 IOCD Byte Count Arguments 5-30
5.5.3 Test STAR Specification Arguments 5-30
5.5.4 Load Mode Register Arguments 5-31
5.5.5 Track and Sector Flag Byte Arguments 5-32
5.5.6 Access Control Flag Byte Arguments 5-34
5.6 Parameter Descriptions 5-34
5.7 Disc Drive Default and Test Parameters 5-36
5.7.1 System Disc Flag is Set ON, Drive is in Single Port Mode 5-37
5.7.2 System Disc Flag is Set OFF, Drive is in Single Port Mode 5-38
5.7.3 System Disc Flag is Set ON, Drive is Set to Dual Port by DUAL1 Directive 5-39
5.7.4 System Disc Flag is Set ON, Drive is Set to Dual Port by DUAL2 Directive 5-40
5.7.5 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL1 Directive 5-41
5.7.6 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL2 Directive 5-41
5.7.7 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL0 Directive 5-42

6 I/O COMMAND TRACE

6.1 Tracing I/O Activity 6-1
6.2 I/O Command Trace 6-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould RPU Disc Processor Diagnostic (RP.DPD.r) is a Level Two program which loads and executes under control of the Diagnostic Executive Program (DEXP). The DEXP provides a multiprogrammed environment allowing simultaneous execution of any number of Level Two diagnostics (limited to 512K bytes of memory) while providing direct operator control for fault isolation and troubleshooting.

The diagnostic contains a sequence of 40 tests, which verify the correct operation of a Disc Processor function. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The diagnostic requires a minimum of operator intervention. Those tests requiring operator intervention may be skipped.

Normal operation of the Gould RPU Disc Processor Diagnostic assumes that the media has been formatted by the Gould RPU Media Verification Program. This revision of the diagnostic and the designated operating system both use the Deallocation Track Table to identify defective tracks and to establish the cylinders which the diagnostic program may access and use without destroying the operating system.

If the media has not been formatted by the Gould RPU Media Verification Program, refer to Initial Drive Checkout and Media Formatting, for the procedure establishing the media for diagnostic use.

During execution the diagnostic writes milestone messages and error message sequences to the listed output (LO) file which is assigned to the operator's console or a line printer.

The diagnostic's peripheral language compiler and run-time support package provide:

- The ability to create new directive test sequences.
- The ability to reduce diagnostic test sequences into their failing components for troubleshooting.

The diagnostic reads operator-composed directive sequences from the command control (CC) file (if in batch mode) or from the operator communications (OC) file (if in interactive mode). The CC file is assigned to disc or tape and the OC file is assigned to the operator's console or a terminal configured on an Eight-Line Asynchronous Controller.

1.2 Resource Requirements

The Gould RPU Disc Processor Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory.
- A Disc Processor (Model 8055), or
  A Universal Disc Processor (Model 8060), or
  A Cache Disc Accelerator (Model 8160, 8161, 8162, 8180, 8181, 8182).

One or more of the following disc drives must be configured in the system:

Fixed Head Disc Drive (5 MB) Model 8104

Cartridge Module Disc Drive (32 MB) Model 8117

Winchester Disc Drive (80 MB) Model 8122

Winchester Disc Drive (160 MB) Model 8127

Nine Inch Cartridge Disc Drive (80 MB) Model 8136

Nine Inch Winchester Disc Drive (160 MB) Model 8146

Winchester Disc Drive (675 MB) Model 8155

Eight Inch Winchester Disc Drive (80 MB) Model 8420

Eight Inch Winchester Disc Drive (167 MB) Model 8424

Nine Inch Winchester Disc Drive (340 MB) Model 8858

Removable Disc Drive (80 MB) Model 9342

Removable Disc Drive (300 MB) Model 9344

Note: The Model 8155 (675 MB) disc is analogous to the Model 8155 (600 MB) disc.

CHAPTER 2

ASSIGNMENTS AND INITIALIZATION

2.1 Disc Processor Controller Model Assignment

The diagnostic program defaults to Disc Processor Model 8055. For the Universal Disc Processor (Model 8060), the ..,CON,.. directive is used as follows:

```
DXP> taskname CON8060
```

This conditions the program to perform tests only on commands available for that model number.

2.2 System Disc Flag Assignment

The System Disc Flag Set to ON enables the diagnostic to write only on cylinders determined by the model number or by information obtained from the Deallocation Track Table. This setting enables the diagnostic to read labels, randomly seek and read labels over the entire media but confines write/read tests to the diagnostic cylinders. This capability allows a drive to be tested without destroying the integrity of the labels or data on the media other than the diagnostic tracks. Refer to the ..,SDF,.. Unique Directive for a complete description.

Each drive being tested has its own System Disc Flag which enables complete media or diagnostic testing for each drive. For a description of the test parameters by model number, refer to Disc Drive Test Parameters.

Note: The System Disc Flag Directive should always precede the dual port directives: ..,DUAL1,.., ..,DUAL2,.., ..,DUAL0,.. and the single port ..,SING,.. directive.

2.3 Formatted Drive Assignment

This revision of the Gould RPU Disc Processor Diagnostic and the designated operating system assumes that the media has been formatted by using the Gould RPU Media Verification Program. The Gould RPU Media Verification Program establishes a pointer in cylinder zero, head zero, which is used as a coordinate to access a Deallocation Track Table that contains the reason code for deallocation and the starting and ending absolute binary coordinates of tracks are deallocated. The diagnostic program establishes the diagnostic test cylinders from the entry in the Deallocation Track Table and confines Write/Read Operations to those cylinders if the System Disc flag is 'ON'.

If the System Disc flag is 'OFF', the ending test parameter is up to but not including the Media Descriptor Boundaries (the cylinder that contains the deallocation information).

The diagnostic program operates with the System Disc flag either 'ON' or 'OFF' and does not destroy the pointer in cylinder zero, head zero, or the Deallocation Track Table.

The diagnostic defaults to a single port drive attribute. If the media is to be formatted to dual port, refer to the "D" Unique Directives for the use of the ..,DUAL0,.., ..,DUAL1,.. and ..,DUAL2,.. directives.

2.4 Unformatted Drive Assignment

Media may be formatted by using the ..,ZAP,.. Directive. However, no deallocation track information is written to the media. This directive formats the media but does not check for flaws in sector data. Both this revision of the Gould RPU Disc Processor Diagnostic and the designated operating system function properly if the media is formatted and verified by the Gould RPU Disc Processor Media Verification Program. Therefore, the ..,ZAP,.. directive should only be used as an Initial Drive Checkout and Media Formatting tool to verify that the drive functions properly.

If the ..,ZAP,.. Directive is used or the diagnostic cannot properly identify the track deallocation information, the following message is written to the listed output (LO) file after the diagnostic is started:

```
DIAGNOSTIC WILL USE MODEL # DEFAULT PARAMETERS.
```

2.5 Initial Drive Checkout on Unformatted Media

The diagnostic performs on media that may or may not be formatted.

The diagnostic performs initial drive checkout on unformatted media by performing Tests 1, 2, 3, 5, 6, 7, 8, 10, 11, and 12. These tests perform service interrupt validation, initial drive functions, such as rezero, seek, angular position and sense commands and illegal operations, that check out the drive and controller for the capabilities of providing the proper status and sense information.

If starting the diagnostic on unformatted media or on media that has not been verified by the Gould RPU Media Verification Program, the following message is written to the listed output (LO) file:

```
TRACK LABEL ZERO CANNOT BE ACCESSED FOR DEALLOCATION INFORMATION.
IF MEDIA HAS NOT BEEN FORMATTED USE THE ZAP DIRECTIVE.
```

This indicates that the media is not formatted by the Gould RPU Media Verification Program and the diagnostic program only executes the above-mentioned tests.

The ..,ZAP,.. directive is used for initial drive checkout. However, the diagnostic program does not deallocate the media descriptor, diagnostic test tracks and the hardware test track. The program uses default parameters for the starting and ending parameters for the selected test. The Gould RPU Media Verification Program must be used to properly write the media descriptor information to track label zero. The media descriptor information is used by this revision of the Gould RPU Disc Processor Diagnostic and by the designated operating sytem.

2.6 Media Formatting

No specific test in the diagnostic program has the capability of formatting the entire media. Media Verification is accomplished by executing the Gould RPU Media Verification Program. The program formats the entire media, then verifies the entire media and inserts deallocated tracks into a table that is written to a specific track on the media. Other deallocated tracks are the two diagnostic cylinders, the hardware test track and the cylinder that has the Deallocated Track Table.

Track Label Zero has a pointer in bytes 12 through 15 that is used by the diagnostic program to obtain deallocated track information. This information contains the location of the diagnostic cylinder boundaries. The Gould RPU Media Verification Program determines the deallocated tracks which are used by the Gould RPU Disc Processor Diagnostic and by the designated operating system which deallocates the tracks in the Volume Formatter Program.

2.7 Single Port Assignment

The diagnostic defaults to the single port format. However, if the drive is to be tested in the dual port mode, the media must be formatted as described in the Gould RPU Media Verification Program or by using the ..,ZAP,.. directive. The following section describes dual port testing procedures.

2.8 Dual Ported Operating Procedures

The following operating procedures are used for a dual ported system in the environment indicated. This procedure assumes that the drive has been formatted previously in the dual port mode by using the Gould RPU Disc Media Verification Program that is compatible with the designated operating system.

Environment--One CPU with one disc drive with Disc Processor channels 8 and 9.

	Operating Procedure	Example
1.	Load the Diagnostic Executive Program (DEXP).	LOAD DEXP
2.	Load the RPU Disc Processor Diagnostic program.	LOAD D1 (RP.DPD.r)
3.	Perform the initialization sequence to select the model number, device subaddress and set the dual port condition to DUAL1.	See Initialization Sequence.
4.	Reset the repeat test option flag.	RESET D1,OPT,6,20
5.	Set the bypass operator intervention option.	SET D1,OPT,8
6.	Load the RPU Disc Processor Diagnostic Program using a different taskname than was used in step 2.	LOAD D2 (RP.DPD.r)

7.	Perform the initialization sequence to select the model number, device subaddress and set the dual port condition to DUAL2.	See Initialization Sequence.
8.	Reset the Repeat Test Option flag.	RESET D2,OPT,6,20
9.	Set the bypass operator intervention option.	SET D2,OPT,8
10.	Start the first loaded program.	START D1
11.	Start the second loaded program.	START D2

Environment--Two CPUs with one disc drive with Disc Processor channels 8 and 9.

	Operating Procedure	Example
. Procedure for CP1		
1.	Load the Diagnostic Executive Program (DEXP).	LOAD DEXP
2.	Load the RPU Disc Processor Diagnostic program.	LOAD D1 (RP.DPD.r)
3.	Perform the initialization sequence to select the model number, device subaddress and set the dual port condition to DUAL1.	See Initialization Sequence.
4.	Reset the repeat test option flag.	RESET D1,OPT,6,20
5.	Set the bypass operator intervention option.	SET D1,OPT,8
. Procedure for CP2		
6.	Load the Diagnostic Executive Program (DEXP).	LOAD DEXP
7.	Load the RPU Disc Processor Diagnostic Program.	LOAD D2 (RP.DPD.r)
8.	Perform the initialization sequence to select the model number, device subaddress and set the dual port condition to DUAL2.	See Initialization Sequence.
9.	Reset the repeat test option flag.	RESET D2,OPT,6,20
10.	Set the bypass operator intervention option.	SET D2,OPT,8

11. Start the diagnostic program. START D2

. Procedure for CP1

12. Start the diagnostic program. START D1

Environment--More than one disc drive with one drive single ported and the other drive dual ported with Disc Processor channels 8 and 9.

The procedures described for the first two environments may be used to test more than one drive by assigning more than one subaddress in the Assign command device input and assigning the correct model numbers to the respective subaddresses.

Consider the following:

- Subaddress 0 is a 80 MB disc Model 8055 and is dual ported.
- Subaddress 2 is a Cartridge Module drive Model 8117 and is single ported (never dual ported).

The following changes would have to be made in the procedures for the previously defined environments:

3. Perform the device assignments. ASSIGN D1,DM08,0,2
4. Assign the subaddress with the appropriate model number. DI AU0,MOD9342,AU2,MOD8117
5. Select the high or low partition of the disc to be used by this copy of the diagnostic. DI AU0,ADP,DUAL1

The sequence of diagnostic testing is as follows:

- Subaddress 0 (DRIVE 0) is tested in the Dual Port Mode.
- After completing the Dual Ported Tests, the program loaded as D1 begins to test the cartridge module on device subaddress 2.

2.9 Dual Ported Mode Test Restrictions

Tests 3, 6 through 9, 12, 25, and 37 automatically are skipped if two Gould RPU Disc Processor diagnostic programs are being executed concurrently and the Drive Parameters are established as DUAL1 in one CPU and DUAL2 in the other CPU. The reason is that these tests perform illegal operations (that is, illegal commands, illegal use of commands, illegal label and drive operations and illegal mode register checks).

If these tests are performed, either copy of the diagnostic is expecting a particular type of error condition or specific sense data from the drive or the controller. However, since the programs are operating independently, either copy may report error conditions incorrectly because the other copy has performed an operation that changed the error status condition or sense data.

Test 3, 6 through 9, 12, 25, and 37 are performed if option 20, Enables.

The following is an example of this type of conflict:

1. Copy one of the Gould RPU Disc Processor Diagnostic performs a Seek Out of Bounds command and causes the drive to generate a SEEK ERROR.

2. After completing the SEEK ERROR, copy two is able to access the drive.

3. Copy two performs an illegal command to the controller.

4. Since the drive itself has not been accessed by the diagnostic, the current drive status indicates a SEEK ERROR.

5. However, copy two is expecting a COMMAND REJECT, but an erroneous SEEK ERROR is reported to the listed output (LO) file.

2.10 Diagnostic Initialization

Before initializing the Gould RPU Disc Processor Diagnostic, load it into the system by following the procedures in Chapter 7--Level Two Diagnostics, of the Gould Virtual Memory Diagnostic Facility User's Manual. The program name for loading the diagnostic is RP.DPD.r. After the diagnostic finishes loading, it writes the following initialization sequence to the operator communications (OC) file:

Note: Entering an invalid response to the prompts causes the diagnostic to reissue them.

1. The diagnostic writes the following message:

   ```
   'taskname' INITIALIZATION
   ```

2. The diagnostic then issues the following prompt:

   ```
   DEFAULT CHANNEL ADDRESS=08 ;[CR,?,X,##]?
   ```

   Responding with a carriage return (cr) causes the diagnostic to assign the specified default channel address (08). To modify the default channel address, enter the valid two-character hexadecimal channel address (##). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

   The valid channel address range is 00_{16} through $7F_{16}$.

3. The diagnostic then issues the following prompt:

   ```
   SELECTED NUMBER OF DRIVES=1 ;[CR,?,X,#]?
   ```

   Responding with a carriage return (cr) causes the diagnostic to assign the specified default number of drives (1). To modify the default number of drives, enter the valid one-character decimal number of drives (#). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

   The valid number of drives is from one to eight.

4. The diagnostic then issues the following prompt:

```
SUBCHANNEL INITIALIZATION
DEFAULT DEVICE SUBADDRESS=00;[CR,?,X,#]?
```

Responding with a carriage return (cr) causes the diagnostic to assign the specified default device subaddress (00). To modify the default device subaddress, enter the valid two-character hexadecimal device subaddress (##). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

The valid subchannel addresses are 0_{16} through F_{16} and only even values should be entered.

5. The diagnostic then issues the following prompt:

```
SELECTED MODEL NUMBER=9342 ;[CR,?,X,####]?
ENTRY OF A MODEL NUMBER WILL SELECT THE DRIVE TO
SINGLE PORT.
```

Responding with a carriage return (cr) causes the diagnostic to assign the specified default model number (9342). To modify the default model number, enter the valid four-character decimal model number (####). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

Valid Model Numbers are as follows:

Model #	Cylinder	Heads	Size		Description
8104	64	4	5 MB		Fixed Head
8117	823	2	32 MB	14-in.	Cartridge
8122	823	5	80 MB	14-in.	Winchester
8127	823	10	160 MB	14-in.	Winchester
8136	823	5	80 MB	9-in.	Cartridge
8146	823	10	160 MB	9-in.	Winchester
8155	843	40	675 MB	14-in.	Winchester
8420	823	5	80 MB	8-in.	Winchester
8424	1024	8	167 MB	8-in.	Winchester
8858	711	24	340 MB	9-in.	Winchester
9342	823	5	80 MB	14-in.	Removable
9344	823	19	300 MB	14-in.	Removable

6. The diagnostic then issues the following prompt:

```
SELECTED SECTOR FORMAT=20 ;[CR,?,X,##]?
```

Responding with a carriage return (cr) causes the diagnostic to assign the specified default sector format (20). To modify the default sector format, enter the valid two-character decimal sector format (##). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

Valid sector formats are 16 and 20 sectors per track.

Special Consideration

If the model number is changed during reinitialization the sector format default will be reset at 20.

7. The diagnostic then issues the following prompt:

```
SELECTED PORT CONDITION=SING ;[CR,?,X,TTTT#] ?
```

Responding with a carriage return (cr) causes the diagnostic to assign the specified default port condition (SING). To modify the default port condition, enter the valid port condition (TTTT#). Responding with a question mark (?) causes the utility to write a display of the permissible range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

Valid port conditions are as follows:

SING indicates the entire functional surface or two diagnostic cylinders will be tested using one disc drive in a single system.

DUAL0 indicates the entire functional surface or two diagnostic cylinders will be tested using one disc drive shared by two systems.

DUAL1 indicates the first half of surface or first diagnostic cylinder will be tested.

DUAL2 indicates the second half of surface or second diagnostic cylinder will be tested.

8. The diagnostic then issues the following prompt:

```
IS CACHE DISC ACCELERATOR PRESENT/=N [CR,?,X,Y OR N]
```

Responding with a carriage return (cr) causes the diagnostic to assign the specified default Cache Disc Accelerator answer (N). To modify the default Cache Disc Accelerator answer, enter yes, if it is present (Y). Responding with an X, exits the initialization sequence before its completion.

Responding with Y (yes) causes the diagnostic to write the following message:

```
PUT AUTO/TEST SWITCH IN TEST POSITION TO DISABLE
'CACHE, THEN DEPRESS CDA RESET BUTTON ON FRONT PANEL
```

This procedure should be followed to allow the cache disc accelerator to access the media on the drive rather than using the cache itself.

9. The diagnostic then writes the following messages:

```
USE HELP DIRECTIVE FOR OTHER INFORMATION
INITIALIZATION COMPLETE
```

The first message is only for information. The other message indicates diagnostic initialization is successful. At this time, modify options and/or start the diagnostic.

If the model number is changed during reinitialization the selected port condition default will be reset to SING (single port).

The values last entered during an initialization or reinitialization sequence are the default values for the next initialization or reinitialization.

However, if a model number has been changed during reinitialization, the diagnostic does not follow the "last initialization value entered" rule.

2.11 Diagnostic Option Set

After the Gould RPU Disc Processor Diagnostic loads and initializes, it enters a wait state until receiving directions through the Diagnostic Executive Program (DEXP).

Using the DEXP, select diagnostic options to modify default execution conditions.

The Gould RPU Disc Processor Diagnostic contains the following options:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If the option is set during an error message report, the message is completed. Option 1 is not inhibited by this option.
1	Reset	Rings the bell on the operator's console once per error.
2	Reset	Suspends diagnostic execution if an error is detected. Prior to suspension, the associated error message is written, unless option 0 is set. During suspension, the following message is written: 'taskname' ERROR...;[C,H,T]? The query [C,H,T]? requires a response. Responding with C, continues execution. Responding with T, terminates execution. Responding with H, places the diagnostic on hold, alloting time for setting options, performing memory dumps, or manipulating other diagnostics running concurrently. Nullify the hold state with a Continue command.
3	Reset	Repeats the sequence which detected an error until obtaining a correct result. During the repeat function, no operand values change. If options 0, 1, and 2 are set, their functions occur prior to this function.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the current diagnostic until the option is reset.
7	Reset	Runs only tests coded in directive language.

Options	Default	Effect When Set
8	Reset	Inhibits stand-alone tests which are not performed in a multi-task environment and any test requiring operator intervention.
9	Reset	Inhibits condition code errors and no status interrupt received messages.
10	Set	Writes the diagnostic's pass count message to the listed output (LO) file.
11	Reset	Writes the diagnostic's milestone messages to the listed output (LO) file.
12	Reset	Inhibits both Status and Sense data error message expansion.
13	Reset	Inhibits Status error messages.
14	Reset	Inhibits Sense error messages.
15	Reset	Inhibits Data error messages.
17	Reset	Enables I/O command trace.
18	Reset	Executes the Write Single Sector and Read Single Sector tests.
20	Reset	Conditions the diagnostic to perform Tests 3, 6 through 9, 12, 25, and 37 if the drive is in the dual port mode. This option should be used only if one CPU is active in the dual port mode. Otherwise unexpected errors may be generated by the tests executed through this option.
21	Reset	Inhibits bad track error messages if the track label on cylinder zero, head zero has a pointer to the Deallocation Track Table and the track in error is in the Deallocation Track Table with a reason code of 1 through 8.

2.12 Diagnostic Execution

After setting options, issue a Start command to begin diagnostic execution. As the diagnostic begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-951r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the diagnostic begins testing and writes milestone messages to the listed output (LO) file:

CHAPTER 3

TEST DESCRIPTIONS

3.1 Summary of Tests

The Gould RPU Disc Processor Diagnostic contains 40 tests which run in a continuous sequence, or individually by operator request.

Tests 1 through 11 verify the correct functioning of various Disc Processor functions and exceptional conditions. Tests 12 through 40 verify disc mechanism functions.

Tests 13 through 40 are skipped if the media has not been formatted. The program establishes drive test parameters depending on whether or not a pointer to the Deallocated Track Table has been established in the track label on cylinder zero, head zero of the media by the Gould RPU Media Verification Program.

(A) If track label on cylinder zero, head zero has a parameter in bytes 12 through 15 the program accesses the deallocated tracks and establishes test parameters from the information which is in the deallocation table, (i.e., diagnostic cylinders and end of media testing as is determined by the coordinates in the Media Descriptor Track).

(B) If track label zero is accessed and no parameter is found, the program automatically uses drive default parameters. (Refer to Disc Drive Default and Test Parameters).

The following message is written after a Start directive is issued to the program:

DIAGNOSTIC WILL USE MODEL # DEFAULT PARAMETERS

3.2 Test 1—Interrupt Instruction Verification

This test verifies the following basic interrupt control instructions: Disable Channel Interrupt (DCI), Enable Channel Interrupt (ECI), Activate Channel Interrupt (ACI), Deactivate Channel Interrupt (DACI), Block External Interrupt (BEI), and Unblock External Interrupt (UEI), perform as defined. The test also uses a Test I/O (TIO) instruction to verify that pending interrupts may be processed correctly in conjunction with the interrupt instructions.

This test runs as a stand-alone test and is not designed to perform in a multi-task or system testing mode of operation.

This test may be performed on drives which have formatted or unformatted media.

Note: If both a CPU and an IPU are configured, this test should be run only if the diagnostic program is biased to the CPU by using the OPCOM Bias command.

This test is skipped if option 8 (Inhibits stand-alone tests) is set.

3.3 Test 2--Service Interrupt Halt I/O, Stop I/O, Test I/O Verification

This test verifies that the Disc Processor correctly generates a service interrupt (SI) as a result of a Start I/O Sense and that Sense Data Received from a Rezero command is correct. The test also verifies that the Halt I/O, Stop I/O, and Test I/O instructions function properly.

This test may be performed on drives which have formatted or unformatted media.

3.4 Test 3--Quiescent Device Status Verification

This test verifies that the Disc Processor can perform a Rezero command and detect a Unit Check status and an Intervention Required sense bit (bit 9) set when the device is offline.

This test may be performed on drives which have formatted or unformatted media.

This test is skipped if the drive is in dual port mode and option 20 is reset, or if option 8 (Inhibits operator intervention tests) is set.

This test also verifies that a Channel Program Check status and a Mode Check sense data setting is received if the jumper is absent from the DIA board during execution of a Set Reserve Track Command.

The following message is not applicable if the reserved track jumper has not been inserted in the DIA (device interface adapter) board:

RESERVED TRACK JUMPER #4 SHOULD BE REMOVED

Note: The message is not written if the controller model number is 8055.

3.5 Test 4--Label Verification

This test verifies Label verification in the following two operating modes which are dependent upon the System Disc Flag setting:

- Label verification of the diagnostic cylinders.
- Label verification of the entire media.

3.5.1 Label Verification of the Diagnostic Cylinders

Label verification of the diagnostic cylinders is considered the default mode of operation. The program reads all track and sector labels on the diagnostic cylinders and performs a 30-byte data compare on all the sectors to insure that the label information is correct. Any data errors detected are written to the listed output (LO) file as data errors without any explanation of flag byte errors. All diagnostic track and sector labels are considered to have only a setting of 'GOOD TRACK' in the flag byte.

3.5.2 Label Verification of the Entire Media

Label verification of the entire media is performed if the System Disc Flag is OFF. The program reads all track and sector labels but only compares the first four bytes of data for correct cylinder, head, and sector information. Track and sector Flag byte data other than 'GOOD TRACK' is written to the LO file with the appropriate messages. (Refer to Label Verification Messages.)

The following example shows how to condition the diagnostic for this operating mode.

Consider the following:

- The drive is a model number 8117 (32 MB) connected to device subaddress 6_{16}. The plug on the drive is set to 3.
- The program is loaded with the name D1.
- The fixed disc pack is to have labels verified.
- The sequence starts after the ASSIGN procedure is performed.

```
DXP>RESET D1 OPT 6,19
DXP>D1 AU6,MOD8117,SDF='SYS(ON)',CMF
DXP>START D1 4
```

Note: The ..,CMF,.. directive conditions the diagnostic to verify the fixed media on the drive. The default condition for the Cartridge Module Model 8117 would be to verify the removable media.

3.6 Test 5--Basic Disc Functions

This test is composed of subtests which verify that the Disc Processor correctly performs basic device commands. The Seek commands use the diagnostic cylinder for whatever model device is being used. The test verifies the following commands:

- Rezero
- Angular Position
- Sense
- Seek
- Seek and Increment Head
- Seek and Read (is skipped if track label zero pointer is missing)
- Seek and Read Good Sector Labels (is skipped if track label zero pointer is missing)
- Seek and Read Good Track Label (is skipped if track label zero pointer is missing)
- Seek and Test Sector Target Address Register (STAR) with No Status Modifier
- Seek and Test STAR with Status Modifier

This test may be performed on unformatted drives, however, no read operations will be performed.

3.7 Test 6--Invalid Memory Access

This test verifies that the Disc Processor correctly detects an invalid memory access condition if requested to transfer data into a nonpresent memory module. The controller responds with a Channel Program Check Status Error (bit 3) and an Invalid Memory Address Sense Bit (sense word 1, bit 21).

For systems with 16 MB of memory, this test is automatically skipped following the test identification milestone message because no nonpresent memory module is in the test range.

This test is skipped also, if the drive is in dual port and option 20 is reset.

3.8 Test 7—Channel Program Violation

This test is composed of 11 subtests which verify the detection of channel program violations and drive violations.

The test may be performed on unformatted media.

Subtest	Function
7A.	Performs a Seek command to an out of bounds cylinder and verifies that a Unit Check Status Error, sense bit 14, is set. It also verifies that the sense data bits are set for Disc Addressing Error, bit 25 in sense word 2, and Seek Error, bit 8 is set in sense word 4 for all drives other than the Model 8420 and 8588. The expected sense data is checked for seek error and no unit check status is received from the controller.
7B.	Performs a rezero command and verifies that the device properly repositions itself; that the status returned is channel and device end status.
7C.	Performs an Increment Head command to an out of bounds cylinder and verifies that the Unit Check status bit 14 is set. It also verifies that sense bit 25 in sense word 2 is set, indicating a Disc Addressing Error, and sense bit 8 in sense word 4 is set, indicating a Seek Error.
7D.	Verifies that a Transfer In Channel (TIC) as the first command in an I/O command list (IOCL) sets status bit 3, indicating Channel Program Check, and sets sense bit 8 of sense word 2, indicating a Command Reject is detected.
7E.	Verifies that a Transfer In Channel (TIC) to a TIC sets status bit 3, indicating a Channel Program Check, and sets sense bit 8 in sense word 2 which indicates a Command Reject is detected.
7F.	Verifies that a TIC command cannot have a data address pointing to a byte boundary by setting the Channel Program Check, status bit 3, and the Command Reject, sense bit 8 of sense word 2.
7G.	Verifies that a TIC command cannot have a data address pointing to a halfword boundary by setting the Channel Program Check, status bit 3, and the Command Reject, sense bit 8 of sense word 2.
7H.	Verifies that a Sense command with a byte count of five sets an Illegal Length, status bit 2, and a Channel Program Check, status bit 3.
7I.	Verifies that a Seek command with a byte count of five sets an Illegal Length, status bit 2, and a Channel Program Check, status bit 3.

Subtest	Function
7J.	Verifies that a Data Chain command flag byte setting of a Read Angular Position command sets a Channel Program Check, status bit 3, and a Chaining Error, sense bit 23 of sense word 2.
7K.	Verifies that the device generates a Channel Program Check, Channel End, and a Device End.

This test is skipped if the drive is in dual port mode and option 20 is reset.

3.9 Test 8—Channel Program Violation, Undefined Commands

This test verifies that the Disc Processor correctly detects a channel program violation condition if commanded to perform all undefined commands. The controller responds by setting the Channel Program Check status bit 3, and the Command Reject sense data, bit 8, in sense word 2.

This test may be performed on formatted or unformatted media.

This test is skipped if the drive is in dual port mode and option 20 is set.

3.10 Test 9—Track and Sector Label Flag Verification

This test is composed of 14 subtests which verify that the Disc Processor/device properly performs various track and sector label operations. This test verifies the following: defective track and sector write/read, alternate track sector write/read, set a spare track and related sectors, set a reserved track and related sectors and will also test for various illegal track and sector conditions.

This test is skipped if track label zero, head zero cannot be accessed properly.

This test is skipped if the drive is in dual port mode and option 20 is reset.

Test 9 is skipped if the drive is in Dual Port Mode One or Two (DUAL1 or DUAL2) and System Disc Flag is set ON. The test is executed if the drive is in the DUAL0 mode of operation.

Test 9 is skipped in the dual port mode because the diagnostic performs defective and alternate track tests on both diagnostic cylinders. This test is the only test designed to write/read data to both the DUAL1 and DUAL2 designated diagnostic cylinders instead of using the single designated cylinder.

This test should be executed only by one CPU at a time by setting option 20, then starting the test.

3.11 Test 10—Unsupported Transaction Verification

This test verifies that the Disc Processor correctly detects an unsupported transaction condition code if an attempt is made to execute a Write Channel or Grab Controller instruction.

This test may be performed on drives which have formatted or unformatted media.

3.12 Test 11--Halt I/O, Stop I/O Program Termination

This test verifies that the Disc Processor correctly performs a Halt I/O and Stop I/O instruction after a Start I/O is issued to the device.

This test may be performed on drives which have formatted or unformatted media.

3.13 Test 12--Variable Length Seeks

This test verifies the ability to correctly perform a full range of variable length seeks. The test sequence follows:

1. PB=0,PC=U,..

   Sets the lowest track address coordinate to zero and calculates the highest track address coordinate.

2. ..,1,PT=B,SEK,PT=C,..

   Positions the disc mechanism to the address coordinate contained in parameter B and positions the disc mechanism to the address coordinate contained in parameter C.

3. ..,PB+1,J1/B<=U,:

   Increments the track address coordinate contained in parameter B, decrements the track address coordinate contained in parameter C and repeats the seek sequence until the track address coordinate contained in parameter B is greater than the highest track address coordinate parameter.

This test may be performed on drives which have unformatted media to verify the seek capability of the drive.

This test is skipped if the drive is in dual port mode and option 20 is reset.

3.14 Test 13--Write/Read Diagnostic Track and Highest Testable Track

This test verifies the ability to write to and read from the lowest (track zero) and highest tracks. The test sequence is as follows:

1. PW=V,SH0,SS0,PT=Q,..

   Defaults the transfer count to one sector, selects head zero selects sector zero, selects the diagnostic track.

2. ..,SEK='CC',LFO,WR1,..

   Inserts a Seek command chained into the I/O command list (IOCL) queue, loads output buffer with a floating one data pattern and performs a one sector Write command.

3. ..,SEK='CC',RR1,..

   Inserts a Seek command chained into the IOCL queue and performs a one sector Read command.

4. ..,PT=Q,PH=K,SPH,PS=J,SPS,..

   Selects the diagnostic track address coordinate, selects the highest head address coordinate and selects the highest sector address coordinate.

5. ..,LFZ,SEK='CC',WR1,..

   Loads the output buffer with a floating zero data pattern and performs a Seek command chained to a Write command.

6. ..,SEK='CC',RR1,STZ,:

   Performs a Seek command chained to a Read command and verifies the data read and performs a Rezero command.

3.15 Test 14—Write/Read Zeros, Ones, Floating Zero Data

This test verifies the ability to write and read data patterns of all zeros, all ones and floating zero on the first track of the diagnostic cylinders. The test sequence is as follows:

All Zeros Pattern

1. CML,PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate Q to parameter T, sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LZ,WR,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the output buffer with all zero's data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,:

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, inserts a Read command into the IOCL and issues a Start I/O instruction and verifies the data.

All Ones Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q and sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LO,WR,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the output buffer with all ones data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command in the IOCL and issues a Start I/O instruction, and verifies the data.

Floating Zero Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q, sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LFZ,WR,..

   Inserts a Seek, Command Chained, into the IOCL queue, loads the output buffer with all floating zeros data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,STZ,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command in the IOCL queue and issues a Start I/O instruction, and verifies the data and performs a rezero command to the device.

3.16 Test 15—Write/Read Floating Ones, Binary Progression and Regression

This test is similar to Test 14 and verifies the ability to write and read floating ones, binary progression and binary regression patterns to the first cylinder of the diagnostic cylinders. The test sequence is as follows:

Floating Ones Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q, and sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LBP,WR,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the output buffer with a binary progression data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command into the IOCL and issues a Start I/O and verifies the data.

Binary Progression Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q, and sets the head and sector coordinate for the Seek command to zero.

2. ..,LBP,WR,..

   Loads the output buffer with binary regression data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command into the IOCL and issues a Start I/O instruction and verifies the data.

Binary Regression Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q, and sets the head and sector coordinate for the Seek command to zero.

2. ..,LBP, WR,..

   Loads the output buffer with binary regression data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SK='CC',RR,STZ,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command into the IOCL and issues a Start I/O instruction and verifies the data, then performs a rezero command.

3.17 Test 16—Write/Read Random and ASCII Data

This test is similar to Test 14 and verifies the ability to write and read random and ASCII data patterns. The test sequence follows:

Initial Random Data Pattern

1. SEK='CC',LIR,WR,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue loads the output buffer with initial random data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

2. ..,SEK='CC',RR1,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read Command into the IOCL and issues a Start I/O instruction and verifies the data.

Second Random Data Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter to the diagnostic cylinder parameter Q, and sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LNR,WR,..

   Inserts a Seek, Command Chained, into the IOCL queue, loads the output buffer with new random data pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command into the IOCL and issues a Start I/O instruction and verifies the data.

ASCII Data Pattern

1. PW=V,PR=E,PT=Q,SH0,SS0,..

   Defaults the transfer count to one sector, defaults the record count parameter R to the number of sectors on one head, sets the initial track coordinate parameter T to the diagnostic cylinder parameter Q, and sets the head and sector coordinate for the Seek command to zero.

2. ..,SEK='CC',LA,WR,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the output buffer with 64-character ASCII subset, inserts a Write command into the IOCL queue and issues a Start I/O instruction.

3. ..,SEK='CC',RR1,STZ,:

   Inserts a Seek, Command Chained, into the IOCL queue, inserts a Read command into the IOCL and issues a Start I/O instruction and verifies the data, then performs a rezero command.

3.18 Test 17—Write/Read All Sectors on One Track

This test verifies the functioning of sector addressing by writing to and reading from all sectors of track head zero. The test sequence follows:

1. PW=V,SH0,SS0,PT=Q,PS=0,1,SPS,..

   Defaults the transfer count to one sector, sets the head and sector coordinate to zero, sets the current track coordinate to the diagnostic cylinder, defaults the sector address coordinate variable parameter S to zero, and sets the sector address coordinate contained in parameter S.

2. ..,SEK='CC',LAD,WR1,PS+1,J1/J,..

   Inserts a Seek, Command Chained, into the IOCL and loads the output buffer with the disc address data pattern, inserts a Write command into the IOCL and issues a Start I/O instruction, increments the sector address coordinate contained in parameter S, and repeats the write/increment sequence until all sectors on track head zero have been written.

3. ..,2,SEK='CC',RR1,PS-1,SPS,LAD,J2/J,:

   Inserts a Seek, Command Chained, into the IOCL, inserts a Read command into the IOCL and issues a Start I/O instruction, decrements the address coordinates contained in parameter S and repeats the read/decrement sequences until all sectors on the diagnostic track, head zero, have been read.

3.19 Test 18—Head Addressing

This test verifies the functioning of disc head addressing by writing to and reading from sector zero on all heads of the diagnostic cylinder. The test sequence follows:

1. PW=V,PT=Q,PH=0,SS0,1,SPH,..

   Defaults the transfer count to one sector, selects the diagnostic track, defaults the head address coordinate variable parameter H to head zero, selects the sector zero and selects the head address coordinate contained in parameter H.

2. ..,LAD,WR1,PH+1,J1/K,..

   Inserts a Seek, Command Chained, into the IOCL queue, loads the output buffer with the disc address coordinate pattern, inserts a Write command into the IOCL queue and issues a Start I/O instruction, increments the head address coordinate contained in parameter H, and repeats the seek/write/head increment sequence until sector zero on all heads has been written.

3. ..,2,SEK='CC',RR1,PH-1,SPH,LAD,J2/K,STZ,:

   Then inserts a Read, Command Chained, into the IOCL queue and issues a Start I/O, decrements the head address coordinate parameter H and sets the head coordinate in parameter H, refurbishes the output buffer with the disc coordinates and repeats the seek/read/decrement head sequence until sector zero on all heads has been read, then a rezero command will be issued to the drive to complete the test.

3.20 Test 19—Write Disc One Track Per Start I/O

Test 19 verifies the ability to write to all sectors and head over the selected media using one Start I/O Instruction per Track Write Operation.

This test writes one full track of data per Start I/O instruction, and should always precede the execution of Tests 20 through 24 as the data written during this test are verified during their execution. If a significant number of errors are detected during verification of data written by Test 19, the Single Sector Write-Test 38 should be performed in lieu of Test 19.

The data pattern used by model numbers 8136 and 8146 is the 2,7 RLL pattern, which is 'C67BC67B'. All other model numbers use the '5A5A5A5A' data pattern.

The test sequence follows:

1. PW=V,PW*E,PD=D,PT=L,..

   Defaults the transfer count to one track, loads the output buffer with a fixed data pattern, defaults the track address coordinate parameter T to parameter L.

2. ..,PH=0,2,SPH,PS=0,3,SPS,..

   Defaults the head address coordinate variable parameter H to zero, sets the head address coordinate contained in parameter H, defaults the sector address coordinate variable parameter S to zero, sets the sector address coordinate contained in parameter S.

3. ..,SEK='CC',LID,WR1,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the first word of each sector in the output buffer with the track, head, sector identification pattern inserts a Write command into the IOCL queue and issues a Start I/O instruction.

4. ..,PH+1,J2/K,..

   Increments the head address coordinate variable parameter H by one, and repeats the Seek/Write/head increment sequence until all sectors and heads on track parameter T have been written.

5. ..,PT+1,J1/T>=U,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the seek/write/sector increment/head increment/track increment sequence until parameter T is greater than parameter U.

3.21 Test 20--Read Disc One Track Per Start I/O

Test 20 verifies the ability to read from all sectors and head over the selected media cylinders using one Start I/O Instruction per Track Read Operation.

This test reads one full track of data per Start I/O instruction. Test 19--Write Disc One Track, or Test 38--Write Single Sector should be performed as a prerequisite to insure that the correct data is being compared during the execution of Test 20. If a significant number of errors are detected during the performance of this test, Test 39--Single Sector Read should be performed in lieu of Test 20.

The test sequence follows:

1. PW=V,PW*E,PD=D,PT=L,..

   Defaults the transfer count to one tract, loads the output buffer with a fixed data pattern, and defaults the track address coordinate variable parameter T to parameter L.

2. ..,PH=0,1,2,SPH,PS=0,3,SPS,..

   Defaults the head address coordinate variable parameter H to zero, sets the head address coordinate in parameter H, defaults the sector address coordinate variable parameter S to zero and sets the sector address coordinate in parameter S.

3. ..,SEK='CC',LID,RR1,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the first word of the output buffer with the track, head, sector identification pattern, and inserts a Read command into the IOCL queue and issues a Start I/O instruction.

4. ..,PH+1,J2/K,..

   Increments the head address coordinate variable parameter H by one, and repeats the seek/read/head increment sequence until all sectors and heads on track parameter T have been read.

5. ..,PT+1,J1/T>=U,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the Seek/Read/sector increment/head increment/track increment sequence until parameter T is greater than parameter U and concludes test execution with a rezero command.

3.22 Test 21--Automatic Head Advance

This test verifies automatic head advance through all increment sequences except those heads that cause an automatic track advance.

The performance of Test 19--Write Disc One Track Per Start I/O, or Test 38--Write Single Sector, is a prerequisite to the execution of this test. The data written in Test 19 or 38 are verified by Test 21.

The data pattern used by model numbers 8136 and 8146 is the 2,7 RLL pattern, which is 'C67BC67B'. All other model numbers use the '5A5A5A5A' data pattern.

The test sequence follows:

1. PW=V,PW*5,PD=D,PS=J,..

   Uses the CDP directive to obtain the correct value for parameter B, defaults the transfer count for five sectors, loads the output buffer with a fixed data pattern and defaults the sector address coordinate variable parameter S to the highest sector address.

2. ..,SPS,PT=L,1,SPT,..

   Selects the disc coordinate contained in parameter S, defaults the track address coordinate variable parameter T to parameter L and sets TRKHDSEC to the current contents of the value contained in parameter T.

3. ..,PH=0,2,SPH,..

   Defaults the head address coordinate variable parameter H to zero and selects the head address coordinate contained in parameter H.

4. ..,SEK='CC',LID5,LIZ,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the first word of each sector's portion of the output buffer with the track head, and sector identification pattern and clears the input buffer.

5. ..,RR1,..

   Inserts a Read command into the IOCL queue and issues a Start I/O instruction.

6. ..,PH+1,J2/B,..

   Increments the head address coordinate variable parameter H by one, and repeats the read/head increment sequence until all heads have been automatically advanced except those heads which cause an automatic track advance.

7. ..,PT+1,J1/T>=U,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the read/head increment sequence until parameter T is greater than parameter U and ends the test with a rezero command.

3.23 Test 22--Automatic Head Advance, Data Chained Read

This test verifies data chained automatic head advance through all increment sequences except those heads that cause an automatic head advance.

The performance of Test 19--Write Disc One Track Per Start I/O, or Test 38--Write Single Sector, is a prerequisite to the execution of this test. The data written in Test 19 or 38 is verified by Test 22.

The data pattern used by model numbers 8136 and 8146 is the 2,7 RLL pattern, which is 'C67BC67B'. All other model numbers use the '5A5A5A5A' data pattern.

The test sequence follows:

1. PD=D,PS=J,..

   Uses the CDP directive to obtain the correct value for parameter B, loads the output buffer with a fixed data pattern and defaults the sector address coordinate variable parameter S to the highest sector address.

2. ..,SPS,PT=L,SEK='CC',..

   Sets the current value of parameter S into the disc coordinate, defaults the initial track coordinate to parameter L, inserts a Seek, Command Chained, into the I/O command list (IOCL).

3. ..,PH=0,2,SPH,PW=V,..

   Sets the initial head coordinate in parameter H to zero, sets the head coordinate from parameter H into the current coordinate and defaults the byte count to the value in parameter V.

4. ..,SEK='CC',LID5,RR='DC',RR='DC',RR='DC',RR='DC',..

   Inserts a Seek, Command Chained, into the IOCL, loads the first word of each sector's portion of the output buffer with the track, head, and sector identification pattern and inserts four Data Chained Read commands into the IOCL queue.

5. ..,RR1,PH+1,J2/B,..

   Inserts a Read command into the IOCL queue and issues a Start I/O, increments the head address coordinate variable parameter H by one and repeats the Read/increment head sequence until all heads have been automatically advanced.

6. ..,PT+1,J1/T>=U,:

   Then increments the track address coordinate variable parameter T by one, and repeats the Read/increment head sequence until all heads have been automatically advanced and parameter T is greater than parameter U.

3.24 Test 23—Automatic Track Advance

This test verifies the operation of automatic track advance for all tracks on the disc drive.

The performance of Test 19--Write Disc One Track Per Start I/O, or Test 38--Write Single Sector, is a prerequisite to the execution of this test. The data written in Test 19 or 38 is verified by Test 23.

The data pattern used by model numbers 8136 and 8146 is the 2,7 RLL pattern, which is 'C67BC67B'. All other model numbers use the '5A5A5A5A' data pattern.

The test sequence follows:

1. PW=V,PW*5,PD=D,J2/T=U,PT=L,PH=K,..

   Defaults the transfer count to five sectors, loads the output buffer with a fixed data pattern, jumps to exit two if parameter T equals parameter U sets the track address coordinate variable parameter T to parameter L and defaults the head address coordinate variable parameter H to the highest head address.

2. ..,SPH,PS=J,SPS,..

   Sets the head address coordinate contained in parameter H, defaults the sector address coordinate variable parameter S to the highest sector address and sets the sector address coordinate contained in parameter S.

3. ..,SEK='CC',LID5,LIZ,RR1,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the first word of each sector portion of the output buffer with the track, head and sector identification pattern, clears the input buffer and inserts a Read command with a five sector byte count into the IOCL queue and issues a Start I/O instruction.

4. ..,PT+1,J1/T>=U,2,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the read/increment track sequence until all tracks on the disc drive have been automatically advanced and parameter T is greater than parameter U then performs a rezero command.

3.25 Test 24--Automatic Track Advance, Data Chained Read

This test verifies the operation of data chained automatic track advance on the disc drive.

The performance of Test 19--Write Disc One Track Per Start I/O, or Test 38--Write Single Sector, is a prerequisite to the execution of this test. The data written in Test 19 or 38 is verified by Test 24.

The data pattern used by model numbers 8136 and 8146 is the 2,7 RLL pattern, which is 'C67BC67B'. All other model numbers use the '5A5A5A5A' data pattern.

The test sequence follows:

1. PT=L,J2/T=U,PH=K,SPH,PS=J,SPS,PW=V,PW*5,PD=D,PS=J,SPS,..

   Sets the starting parameter T to parameter L and jumps to directive statement 2 if parameter T equals U sets parameter H to K and inserts the head coordinate into the seek coordinate, sets parameter S to parameter J and inserts the sector coordinate into the seek coordinate, defaults the transfer count for five sectors, loads the output buffer with a fixed data pattern, defaults the sector address coordinate variable parameter S to the highest sector address and sets the sector address coordinate contained in parameter S.

2. ..,PT=L,1,SPT,PH=0,2,SPH,..

   Defaults the track address coordinate variable parameter T to parameter L, positions the disc mechanism to track address coordinate contained in parameter T, defaults the head address coordinate variable parameter H to zero and selects the head address coordinate contained in parameter H.

3. ..,SEK='CC',LID5,LIZ,PW=V,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue, loads the first word of each sector's portion of the output buffer with the track, head, and sector identification pattern, clears the input buffer, and sets the byte count to the total number of bytes in one sector.

4. ..,RR='DC',RR='DC',RR='DC',RR='DC',RR1,PH+1,J2/K,..

   Inserts four data chained Read commands into the IOCL queue followed by a Read command and issues a Start I/O instruction followed by a data compare, increments the head address coordinate variable parameter H by one and repeats the Read/increment head sequence until all heads have been automatically advanced except those heads which cause an automatic track advance.

5. ..,PT+1,J1/T>=U,2,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the read/increment head sequence until all heads on the disc have been automatically advanced and parameter T is greater than parameter U then performs a rezero command.

3.26 Test 25—Mode Register Verification

This test verifies the operation of the reserved track and moving head disc modes. The test sequence follows:

1. J5/Y=1,LMR='CAR(+)',LMR='CAR(-)',..

   Tests for the dual port flag setting and jumps to exit if it is set, performs a positive carriage offset , a negative carriage offset.

2. ..,LMR='RDT(+)',LMR='RDT(-),..

   Performs positive read timing offset negative read timing offset.

3. ..,LMR='RES(ON)',LMR='RES(OFF)',5,:

   Sets reserved track mode ON and then OFF and verifies that the correct status responses follow each command and then exits the test.

Test 25 is not performed if the drive is in dual port mode and option 20 is reset.

3.27 Test 26—Random Access Write/Read or Read Labels

This test performs in two operating modes which are dependent on the setting of the System Disc Flag for the particular subaddress being tested. The two operating modes are as follows:

- System Disc Flag Setting is ON

  The diagnostic does not write any data to the disc, but performs random seek and read sector label commands and verifies that the sector labels have a Good Track format. Any labels which have been set to any other setting are reported as errors. That is, if a Write Protect sector label is detected by the diagnostic it is reported as an error. It should be noted that a list of all labels other than Good Track Format labels is obtained by executing Test 4.

- System Disc Flag Setting is OFF

  The diagnostic performs random Write and Random Read data commands on random tracks over the media as is determined by the Disc Drive Test Parameters. The original data integerity of the media is lost. This mode of operation is used if there is a gross malfunction in the Seek-Write or Seek-Read logic in the drive.

The test sequence follows:

1. 5,LIR,J5/M=1,..

   Initializes the random number and jumps to directive test statement number listed in paragraph 6 if the System Disc Flag Setting is 'ON'

2. PW=V,LIR,1,PB=4,..

   Defaults the transfer count to one sector and initializes the random number generator and loads the output buffer with a random data pattern.

3. ..,2,LRA,SPS,SPH,..

   Calculates a random track, head, and sector address coordinate, and stores the resultant address coordinate values in the track variable parameter T, and head variable parameter H, the sector variable parameter S and sets the current sector and head address to the disc seek table.

4. ..,SEK='CC',WR1,PB+4,J2/49,..

   Inserts a Seek, Command Chained, to a Read command into the I/O command list (IOCL) queue and issues a Start I/O instruction, then adds 4 to Parameter Band and repeats the write sequence until 50 random addresses have been written.

5. ..,PB=R,4,LRA,SPS,SPH,SEK='CC',LIZ,RR1,PB+4,J3/49,LNR,J1/39,J9,..

   Reconstructs the 50 random addresses and reads and verifies the data patterns read. The entire sequence is then repeated until a total of 2000 random addresses have been written and read then jumps to nine which performs a rezero and exits the test entry point if system disc flag is ON.

6. ..,5,PW=30,6,PB=4,7,LRA,SPS,SPH,J8/T=0,..

   Sets the byte count to 30 for sector label operations and sets parameter B equal to four, sets the current sector and head address to the disc seek table, jumps to statement 9 if the track address is a zero.

7. ..,SEK='CC',LLS='GOOD',..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue and conditions the output buffer to Good track and sector label format.

8. ..,LIZ,RLS,..

   Clears the input buffer, inserts a Read Sector Label command into the IOCL queue and issues a Start I/O instruction.

9. ..,PB+4,J7/49,..

   Increments parameter B for the next random number, 8, jumps to statement seven, and repeats the operation 49 times.

10. ..,LNR,J6/49,9,STZ,:

    Fetches a new random number and jumps to statement six, 39 times and concludes the test performing a rezero command.

3.28 Test 27—Write Diagnostic Track Labels

This test verifies the ability to write track and sector labels to the deallocated diagnostic tracks. No other tracks other than the diagnostic tracks are accessed by this test. The test operates independently of the System Disc Flag Setting. The test sequence is as follows:

1. PT=Q,SS0,SH0,PW=E,PW*30,1,PH=0,2,SS0,SPH,3,SP5,..

   Defaults the track address to parameter Q, sets the head and sector address to zero, defaults the sector label byte count to 600 bytes for track sector format 20, or 480 bytes for track sector format 16, sets the initial head and sector value to zero and selects head parameter H and parameter 5.

2. ..,SEK='CC',..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL).

3. ..,WLS='BCPW,GOOD,CC',..

   Inserts a Write Sector Label, Command Chained into IOCL with the sector flag byte modifier of good track.

4. ..,WLT='GOOD',PH+1,J2/K,..

   Inserts a Write Track Label, Command Chained, into the IOCL with a good track flag byte modifier, increments the head count parameter H by one and repeats the Seek/Write sector Labels Write track label until all heads on the device have been tested.

5. ..,J1/T>=U,..

   Then increments the track address and repeats the seek/write sector labels until write track label parameter T is greater than the last allowable track address on the pack.

6. ..,STZ,:

   Issues a Rezero command to the drive.

3.29 Test 28---Read Diagnostic Track Labels

This test verifies the ability to read all track and sector labels on the deallocated diagnostic cylinders. The test sequence follows:

1. PW=E,PW*30,PT=Q,SS0,SH0,3,..

   Defaults the sector label byte count to 600 bytes for track sector format 20 or 480 bytes for track sector format 16, defaults the track address to parameter Q, and defaults the starting sector and head to zero.

2. ..,PH=0,4,SPH,..

   Sets parameter H to zero and selects head parameter H.

3. ..,SEK='CC',LLS='BCPW',LLT,LIZ,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) fabricates one track of sector labels in the output buffer, fabricates one track label of data into the output buffer, and clears the input buffer of parameter W bytes.

4. ..,RLS='CC,BCPW',RLT,..

   Inserts a Read Sector Label, Command Chained, into IOCL queue followed by a Read Track Label command and issues a Start I/O instruction, after which a data compare is made of the fabricated output buffer and the track and sector labels read from the disc.

5. ..,PH+1,J1/K,..

   Increments the read count parameter H by one and repeats the Seek, Command Chained/Read Sector Labels, Command Chained/read track label sequence until all heads on the track have been tested.

6. ..,PT+1,J3/T>=U,STZ,:

   Increments the track address and repeats the sequence until parameter T is greater than the last allowable track address on the pack and ends the test with a rezero command.

3.30 Test 29--Write Data Chained--Four Sectors/Start I/O

Test 29 is a prerequisite test for the execution of Test 30. The test sequence follows:

1. PT=L,STZ,1,PH=0,2,PS=0,SPH,..

   Sets the initial track to parameter L and performs a Rezero command sets the current head and sector parameter to zero sets the current head address by using the contents of parameter H.

2. ..,PW=V,PW*4,LBP,PW=V,..

   Prepares to load the output buffer with four sectors of a binary progression and sets the value for the byte count to the default byte count contained in parameter V.

3. ..,SEK='CC',..

   Inserts a Seek, Command Chain, into the I/O command list (IOCL) queue.

4. ..,WR='DC',WR='DC',WR='DC',..

   Inserts three Write, Data Chained, into the IOCL queue.

5. ..,WR1,PB+4,PS+4,J3/B=J,PH+1,J2/K,..

   Inserts a Write command into the IOCL and issues a Start I/O instruction, increments the module counter, increments the sector parameter by four, continues the seek/write data chained sequence until the value of parameter B equals J, increments the head parameter H and continues until all heads have been written.

6. ..,J1/T>=U,STZ,:

   Increments the track parameter T and continues the seek/write data chained/ increment head/increment track sequence until parameter T is greater than parameter U, rezeros the drive and exits the test.

3.31 Test 30--Read Data Chained - Four Sectors/Start I/O

Test 30 verifies that the data written Test 29 is valid. This test performs a Seek Command Chained to a four-sector data-chained read sequence to verify the data.

Test 29--Write Data Chained--must be performed before the execution of Test 30 because the data verification is based on Test 29 execution.

The test sequence follows:

1. PT=L,STZ,1,PH=0,2,PS=0,SPH,..

   Sets the initial cylinder to parameter L and performs a rezero command sets the current head and sector parameter to zero, sets the current head address by using the contents of parameter H.

2. ..,PW=V,PW*4,PB=0,3,SPS,LBP,PW=V,..

   Prepares the byte count for a four sector operation and sets parameter B to zero, sets the current sector coordinate to the current value of parameter S and conditions the output buffer with a binary progression and sets the value of the byte count to the default parameter value V.

3. ..,SEK='CC',RR='DC',RR='DC',RR='DC',RR1,..

   Inserts a Seek, Command Chained, into the I/O command list (IOCL) queue inserts three Read, Data Chained, commands into the IOCL followed by a Read command, and issues a Start I/O instruction.

4. ..,PB+4,PS+4,J3/B>=J,..

   Increments the module counter increments the sector parameter by four and continues the seek/read data chained sequence until the value of parameter B is greater than J.

5. ..,PH+1,J2/K,..

   Increments the head parameter H and continues to three for head parameter K times.

6. ..,PT+1,J1/T>=U,STZ,:

   Increments the track parameter T and continues the seek/read data chained/increment head/increment track sequence until parameter T is greater than parameter U, performs a rezero command, and exits the test.

3.32 Test 31—Variable Byte Count Zero Fill

Test 31 verifies the ability to zero-fill a sector with variable byte counts. The test sequence follows:

1. PB=V,PB*2,SS0,SH0,1,PW=V,PW*2,LZ,PT=Q,..

   Defaults the initial decrementing write byte count to two sectors, selects sector and head zero sets the byte count to two sectors and loads the output buffer with all zeroes and selects the diagnostic cylinder.

2. ..,SEK='CC',WR1,..

   Inserts a Seek, Command Chained, to a Write command into the I/O command list (IOCL) queue which will cause the IOCL queue to be executed.

3. ..,LO,SEK='CC',PW=B,WR1,..

   Loads the output buffer with all ones, inserts a Seek command into the IOCL queue, sets the word count to the current decrementing word count and inserts a Write command of parameter B bytes into the IOCL queue and causes a Start I/O instruction to execute the contents of the IOCL queue.

4. ..,PW=V,PW*2,LZ,PW=B,LO,..

   Sets the current word count to two sectors, sets the output buffer to zeros, sets the current word count to the decrementing word count and loads the output buffer with ones for the number of bytes specified in parameter B.

5. ..,PW=V,PW*2,LIZ,SEK='CC',RR1,PB-1,J1/B>=4,STZ,:

   Sets the word count to two sectors, clears the input buffer, inserts a Seek, Command Chained to a Read command, starts the I/O on the IOCL queue, compares the data for correct zero fill write and decrements the decrementing byte count in parameter B, continuing the above sequence until the byte counting in parameter B equals four, then issues a rezero command and exists the test.

3.33 Test 32--Variable Byte Count Two Sector Read

The test sequence follows:

1. PW=V,PW*2,SH0,SS0,PT=Q,..

   Defaults the byte count to two sectors, sets the head and sector zero and sets the current disc coordinate parameter T to the diagnostic cylinder.

2. ..,SEK='CC',LA,..

   Inserts a Seek, Command Chained, command into the I/O command list (IOCL) queue and loads the output buffer with ASCII data.

3. ..,WR='CC',SEK='CC',..

   Inserts a Write, Command Chained, and a Seek, Command Chained, into the IOCL queue.

4. ..,RR1,1,PW-1,..

   Inserts a Read command with parameter W bytes into the IOCL queue, performs a Start I/O instruction on the contents of the queue and decrements the word count in parameter W by one.

5. ..,SEK='CC',LIZ,RR1,J1/W>=4,STZ,:

   Inserts a Seek, Command Chained into the IOCL queue, clears the input buffer and inserts a Read command of parameter W bytes into the IOCL queue, executes a Start I/O instruction compares the data, continues until the byte count of parameter W is less than four, then performs a rezero command and exits the test.

3.34 Test 33—Read Variable Byte Count Data Chained

The test sequence follows:

1. PT=Q,SS0,SH0,PW=V,PW*2,LO,..

   Selects the diagnostic track, sector zero and head zero, defaults the byte count to two sectors and sets the output buffer to all ones.

2. ..,SEK='CC',W1,PB=V,1,PW=V,PW/2,SEK='CC',..

   Inserts a Seek, Command Chained, to a Write command into the I/O command list (IOCL) queue, sets the initial value of parameter B to the number of bytes per sector, sets parameter W to the current number of bytes per sector, and divides the current byte count by two, inserts a Seek, Command Chained, command into the IOCL queue.

3. ..,LIZ,RR='DC',PW=B,RR,..

   Clears the input buffer and inserts a Read, Data Chained of parameter W bytes into the IOCL queue, sets parameter W equal to parameter B, uses the new value of parameter W and inserts the read command into the IOCL queue, initiates a Start I/O instruction and compares the data input upon completion of the data transfer.

4. ..,PB-1,J1/B<=192,STZ,:

   Decrements the byte count in parameter B by one and continues Seek command chain/read data chain/read/decrement read byte count until the value of parameter B is less than four bytes and then performs a rezero command and exits the test.

3.35 Test 34--Write Variable Byte Count Data Chained

The test sequence follows:

1. DMN,PT=Q,SS0,SH0,PB=V,PB*2,PW=V,PW*3,LO,..

   Outputs the diagnostic message at the beginning of the test, selects the diagnostic track, sector zero, head zero and defaults the byte count to two sectors, sets the byte count in parameter W to the number of bytes per sector and multiplies the byte count by three, loads the output buffer with parameter W bytes of ones.

2. ..,SEK='CC',WR1,LO,PW=B,..

   Inserts a Seek, Command Chained, to a Write command into the I/O command list (IOCL) queue and executes a Start I/O instruction on the contents of the queue, loads the output buffer with ones and sets the current byte to the value contained in the decrementing byte count parameter.

3. ..,SEK='CC',WR='DC',PW=V,..

   Inserts a Seek, Command Chained, command into the IOCL list, inserts a data chained Write command of parameter B bytes into the IOCL queue, then sets the byte count to the default number of bytes per sector.

4. ..,WR='CC',PW=V,PW*3,LIZ,LZ,PW=V,PW+B,LO,..

   Inserts a command chained Write command into the IOCL queue and sets the new byte count to the three sectors, loads the input and output buffer with three sectors of zero, sets the current byte count to the default number of bytes per sector and adds to parameter W the current value of the decrementing byte parameter B, loads ones into the number of bytes in the current value of parameter W.

5. ..,SEK='CC',PW=V,PW*3,RR1,PB-1,J1/B=<4,STZ:

   Inserts a Seek, Command Chain, command into the IOCL queue and sets the current byte count of parameter W to three sectors of data, inserts a Read command into the IOCL queue and initiates a Start I/O instruction, decrements the byte count in parameter B by one and continues the sequence to statement one, until the value of parameter B is less than four after which a rezero command will be issued to the drive.

   Note: If system is CPU bound, data over/under runs may occur.

3.36 Test 35--Read, Decrementing Byte Count, Data Chained

The test sequence follows:

1. PT=Q,SH0,SS0,PB=V,PB*2,PW=V,PW*3,LBP,..

   Selects the diagnostic track, head zero, and sector zero and defaults the variable byte count to two sectors and sets the output buffer with three sectors for a forward binary progression.

2. ..,SEK='CC',WR1,1,PS=V,PW*3,LIZ,PW=B,..

   Inserts a Seek, Command Chained, to a Write command into the I/O command list (IOCL) queue and executes a Start I/O instruction on the contents of the queue, defaults the byte count to three sectors and fills the input buffer with zeros, defaults the current byte count to the variable byte count.

3. ..,SEK='CC',RR='DC',PW=V,..

   Inserts a Seek, Command Chained, into the IOCL queue inserts a Read data chained command into the IOCL queue and sets parameter W to parameter V.

4. ..,RR1,PB- 1,J1/B<=4,:

   Inserts a Read command into the IOCL queue and executes a Start I/O instruction, subtracts one from the variable byte count and continues this sequence until the variable byte count is less than four.

3.37 Test 36--Read, Decrementing Byte Count, Skip Data Chain

The test sequence follows:

1. PT=Q,SH0,SS0,PB=V,PB*2,PW=V,PW*3,LBP,..

   Selects the diagnostic track, head zero and sector zero and defaults the variable byte count to two sectors and sets the output buffer with three sector bytes of a forward binary progression.

2. ..,SEK='CC',WR1,1,PW=V,PW*3,LIZ,PW=B,..

   Inserts a Seek, Command Chained, to a Write command into the I/O command list (IOCL) queue, then inserts a Write command into the IOCL queue and executes a Start I/O instruction on the contents of the queue, defaults the byte count to three sectors and fills the input buffer with zeros, defaults the current byte count to the variable byte count.

3. ..,SEK='CC',RR='DC,SKIP',PW=V,..

   Inserts a Seek, Command Chained, to a Read, Data Chained, Skip, command into the IOCL queue and sets the current byte count to the default sector byte count.

4. ..,RR1,PB-1,J1/B=<4,STZ,:

   Inserts a Read command into the IOCL queue and executes a Start I/O instruction, subtracts one from the variable byte count and continues this sequence until the variable byte count is less than four, whereby the test will rezero the drive and exit the text.

3.38 Test 37--Error Correction Code

This test verifies that the Disc Processor Device Model 8055 and 8060 or the cache disc accelerator corrects polluted data in a sector. The program contaminates data by setting the mode register to the diagnostic mode, and writes and reads incorrect ECC's using a byte count of one sector and four bytes. The cache disc accelerator, if being tested, uses special write and read commands and uses the automatic ECC correction of the contaminated data to determine that the automatic ECC correction is functioning properly.

This test begins with the last sector and decrements to the first sector, and then verifies that the ECC error either automatically is corrected by the cache disc accelerator or that the Read ECC error correction command provides the correct displacement and error mask to correct the data if testing Models 8055 and 8060.

The following message is written if the program is testing the cache disc accelerator:

CDA MUST BE IN TEST MODE (CACHE DISABLED)

If the cache disc accelerator is not disabled, erroneous errors are detected by the program since data is written and read from cache disc accelerator instead of being written on the media.

This test is skipped if the drive is in dual port mode and option 20 is reset.

3.39 Test 38--Write Single Sector

This test performs single sector write commands on the designated media, controlled by the System Disc Flag setting. Test 38 should be used if Test 19--Write Disc One Track Per Start I/O--produces an abundance of media errors. As Test 19 attempts to write data to a full track, if an error is detected prior to the completion of a full track transfer, the sectors beyond where the error has occurred will not have any data written to them. This test allows the diagnostic to access and write data to each sector. This test allows the diagnostic to access each sector within the parameters selected to be tested and should be a prerequisite to Test 39.

The default condition for this test is that it is skipped by the diagnostic and is executed only if option 18 is set. The test sequence follows:

1. PW=V,PD='5A5A5A5A',PT=L,..

   Defaults the transfer count to one sector, loads the output buffer with a fixed data pattern and defaults the track address coordinate parameter T to parameter L.

2. ..,SEK,PH=0,2,SPH,PS=0,3,SPS,..

   Inserts a Seek command into the I/O command list (IOCL) queue and executes a Start I/O instruction, defaults the head address coordinate variable parameter H to zero, sets the head address coordinate contained in parameter H, defaults the sector address coordinate variable parameter S to zero and sets the sector address coordinate contained in parameter S.

3. ..,SEK='CC',LID1,..

   Inserts a Seek, Command Chained, into the IOCL queue, loads the first word of the output buffer with the track, head, sector ID pattern.

4. ..,WR1,PS+1,J3/J,..

   Inserts a Write command into the IOCL queue and issues a Start I/O instruction increments the sector count by one and continues the Seek/Write/increment sequence until all sectors of head parameter H have been written.

5. ..,PH+1,J2/K,..

   Increments the head address coordinate variable parameter H by one, and repeats the Seek/Write/increment sector/head increment sequence until all sectors and heads on track parameter T have been written.

6. ..,PT+1,J1/PT=U,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the seek/write/increment sector/head increment/track increment sequence until parameter T is greater than parameter U followed by a rezero command and exits the test.

3.40 Test 39--Read Single Sector

The normal prerequisite test for the execution of this test is Test 38--Write Single Sector. The test performs single sector read commands on the designated media, controlled by the System Disc Flag setting. This test should be used if Test 19--Write Disc One Track Per Start I/O, and Test 20--Read One Track Per Start I/O-- produce an abundance of media errors. The test allows the diagnostic to access each sector and read data within the controlling parameters.

The default condition for this test is that it is skipped by the diagnostic and is executed only if option 18 is set. The test sequence follows:

1. PW=V,PD='5A5A5A5A',PT=L,..

   Defaults the transfer count to one sector, loads the output buffer with a fixed data pattern and defaults the track address coordinate variable parameter T to parameter L.

2. ..,1,SEK,PH=0,2,SPH,PS=0,3,SPS,..

   Inserts a Seek command into the I/O command list (IOCL) queue and executes a Start I/O instruction, defaults the head address coordinate variable parameter H to zero, sets the head address coordinate contained in parameter H, defaults the sector address coordinate variable parameter S to zero and sets the sector address coordinate contained in parameter S.

3. ..,SEK='CC',LID1,..

   Inserts a Seek, Command Chained, into the IOCL queue and loads the first word of the output buffer with the track, head, sector ID pattern.

4. ..,RR1,PS+1,J3/J,..

   Inserts a Read command into the IOCL queue and issues a Start I/O instruction, increments the sector count by one, and continues the Seek/Read/increment sequence until all sectors on head parameter H have been read.

5. ..,PH+1,J2/K,..

   Increments the head address coordinate variable parameter H by one, and repeats the seek/read/increment sector/head increment sequence until all sectors and heads on track parameter T have been read.

6. ..,PT+1,J1/T<=U,STZ,:

   Increments the track address coordinate variable parameter T by one, and repeats the seek/read/increment sector/head increment/track increment sequence until the parameter T is greater than parameter U upon which a rezero command will be issued and exit the test.

3.41 Test 40--Reserve/Release Drive Verification

This test verifies the explicit and implicit Reserve and Release commands.

Test 40 is skipped if the Disc Processor Controller Model 8055 is tested by the program.

The test performs a Reserve command to the drive followed by a Seek, Command Chained to a Write command. The diagnostic then performs a Seek, Command Chained, to a Read command. After completing the read command, the test performs an explicit release and compares the data pattern for errors.

CHAPTER 4

ERROR REPORTING

4.1 Error Message Assignment

The Gould RPU Disc Processor Diagnostic writes all error messages to the listed output (LO) file. The default LO file assignment is to the operator's console.

To assign the LO file to a device other than the operator's console, enter:

```
DXP>ASSIGN LO=ttaa [,dsa]
```

ttaa — specifies the four-character device mnemonic containing the two-character device type code and the two-character hexadecimal channel address.

dsa — specifies an optional device subaddress if the assignment is being made to a magnetic tape device.

4.2 Error Messages and Message Sequence

An error message sequence consists of:

1. The message separation banner

```
<<<<'taskname'----hh:mm:ss>>>>
```

taskname — specifies the one to eight character diagnostic name as indicated in the load map.

hh:mm:ss — specifies the system elapsed execution time or the current time-of-day (if initialized), at the point of error.

2. The error message header

```
TST:ttt, CA:cc, DSA:uu, PC:ppppp
DRTV xxxx
EXP COOR yyy--yy--yy STAR zzz--zz--zz
CMND iiiiiiii IOCD dddddddd dddddddd
```

ttt — specifies the number of the erring test.

cc — specifies the assigned hexadecimal Disc Processor channel address.

uu — specifies the assigned hexadecimal device subaddress.

ppppp — specifies the hexadecimal program counter value within the erring test.

xxxx	specifies the last directive processed before the error is detected.
yyy--yy--yy	specifies the track, head, and sector disc coordinates as calculated by the diagnostic.
zzz--zz--zz	specifies the track, head, and sector disc coordinates as returned in sense word one. This value is the current contents of the Sector Target Address Register (STAR).
iiiiiiii	specifies the current instruction.
dddddddd	specifies the erring I/O command doubleword (IOCD).

3. The status error message

```
STATUS ERROR
EXP (UNIT)u (IOCD) bbbbbb (BITS) eee (COUNT) tttt
ACT (UNIT)u (IOCD) bbbbbb (BITS) eee (COUNT) tttt
```

EXP	indicates the expected status values.
ACT	indicates the actual status values.
u	specifies the hexadecimal device subaddress.
bbbbbb	specifies the address of the expected and actual data address for the I/O command list termination.
eeee	specifies status of the drive.
tttt	specifies the residual hexadecimal byte count from the last I/O command doubleword executed in the I/O command list queue.

3A. One of the following messages is written for each occurrence of a status bit in either the expected or actual status unless option 12 (Inhibits error message expansion) is set. If the message is preceded by a plus (+) sign, the status bit indicated is received but not expected; if preceded by a minus (-) sign, the status bit indicated is expected but not received.

Status Bit	Explanation
0	HIO and STOP FUNCTION
1	PROGRAM CONTROLLED INTERRUPT
2	INCORRECT LENGTH
3	CHANNEL PROGRAM CHECK
4	CHANNEL DATA CHECK
5	CHANNEL CONTROL CHECK
6	INTERFACE CHECK
7	CHAINING CHECK
8	DEVICE BUSY
9	STATUS MODIFIER
10	N/A
11	N/A
12	CHANNEL END
13	DEVICE END
14	UNIT CHECK
15	UNIT EXCEPTION

3B. Sense Buffer Register (SBR, SNS Bytes 5, 6, 7)

One of the following messages is written for each occurrence of a sense word two data bit in either the expected or the actual status unless option 12 (Inhibits error messages) is set. If the message is preceded by a plus (+) sign, the sense data bit indicated was received but not expected, and, if preceded by a minus (-) sign, the sense data bit indicated is expected but not received.

Sense bits 0 through 7 of the sense word two contain the current setting of mode register.

Sense Bit	Explanation
8	COMMAND REJECT
9	OPERATOR INTERVENTION REQUESTED
10	N/A
11	EQUIPMENT CHECK
12	DATA CHECK
13	DATA OVERRUN OR UNDERRUN
14	DISC FORMAT ERROR
15	DEFECTIVE TRACK ENCOUNTERED
16	LAST TRACK FLAG ENCOUNTERED
17	ALTERNATE TRACK FLAG ENCOUNTERED
18	WRITE TO WRITE PROTECTED DISC
19	WRITE LOCK VIOLATION
20	MODE CHECK
21	INVALID MEMORY ADDRESS
22	RELEASE FAULT
23	CHAINING ERROR
24	LOST REVOLUTION
25	DISC ADDRESSING OR SEEK ERROR
26	BUFFER CHECK
27	ECC ERROR IN SECTOR LABEL
28	ECC ERROR IN DATA
29	ECC ERROR IN TRACK LABEL
30	RESERVED TRACK ACCESS ERROR
31	UNCORRECTABLE ECC ERROR

3C. Drive Related Status (SNS Bytes 12, 13)

0	SEEK END
1	UNIT SELECTED
2-7	ANGULAR POSITION COUNTER
8	FAULT
9	SEEK ERROR
10	ON CYLINDER
11	UNIT READY
12	WRITE PROTECTED
13	BUSY

4. The data error message:

```
DATA ERROR(S) FR#: BFO: BFI:
fffff    dd    dd
  .      .     .
  .      .     .
  .      .     .
fffff    dd    dd
xxxxx OUT OF yyyyy
```

FR# indicates the frame number(s) in error.

BFO indicates the output buffer (expected) value(s).

BFI indicates the input buffer (actual) value(s).

fffff specifies the frame number of the first frame in error and each successive error up to five frames in error.

dd specifies the data value in the indicated buffer segment.

xxxxx specifies the total number of frames in error.

yyyyy specifies the total number of frames compared.

Note: The total number of frames compared is equal to the number of bytes specified in the initial transfer count (ITC). However, if the residual bytes count (RBC) indicates that fewer than the requested number of transfers are in fact performed, only the number of actually transferred frames is compared.

4.3 Status Error Report Exceptions

Status word checking is normally performed after all Start I/O (SIO) instructions are completed. If any discrepancies are detected between the expected status and the actual status obtained, the status error(s) is reported. However, the status words are not checked if any of the following error conditions are detected after issuing a Start I/O instruction:

- Condition Code Errors
  If any condition code errors occur, they are reported with no further status checking.

- No Interrupt or Device End Received Error
  If an expected interrupt is not detected from the device, the no status interrupt message is reported with no further status checking.

- Status Address Word Errors
  If the status address is not correct the program does not check the location for valid status information.

4.4 Condition Code Errors

Condition code errors are reported if they are detected during execution of an Extended I/O Instruction. Normal condition codes of request accepted and queued, hexadecimal eight or status stored hexadecimal four, are not reported. The following messages are written if a condition code error is detected:

If a condition code of hexadecimal zero is detected and the current instruction being executed is not a Reset Channel instruction, the following message is written:

```
CC=0, UNASSIGNED
```

This condition code is valid only for the Reset Channel instruction.

If the channel times out and remains busy without allowing for a request accepted or status stored condition, the following message is written:

```
CC=1, CHANNEL BUSY
```

If the channel/controller is not defined, the following message is written:

```
CC=2, UNDEFINED
```

If the subchannel times out as busy and does not allow for a request accepted or status stored conditions, the following message is written:

```
CC=3, SUBCHANNEL BUSY
```

If the Disc Processor channel is rejecting the current command as not being supported by the device, the following message is written:

```
CC=5, UNSUPPORTED
```

If an unassigned condition code is detected, the following message is written:

```
CC=x, UNASSIGNED
```

x specifies hexadecimal 6,7,9,10,11,12,13,14,15

These condition codes are unassigned and should never be received.

If any of the above condition code errors are reported, no attempt is made to check the status information, because there should be no status information received from the Disc Processor.

4.5 No Interrupt and Device End Messages

If an interrupt is expected and not received from the device, or if no device end is reported after an interrupt is received from the device, one of the following messages is written.

If no interrupt is received from the device but an interrupt is expected, the following message is written:

```
CC=a, NO SI RECEIVED
```

a specifies the condition code.

This error message usually follows a condition code error message indicating that the instruction is never executed because the RPU/Controller is not able to accept and/or queue the particular instruction.

If an interrupt is received from the device, but no device end is received as a status condition, the following message is written:

```
CC=a, NO DEVICE END
```

a specifies the condition code.

If this error message is followed by a Status Address Error Message, then the status information is posted incorrectly.

If either of the above messages are reported, no status information or sense data information is checked for validity.

4.6 Status Address Word Error

Status addresses obtained in the service interrupt receiver area are checked for validity with respect to the data address used in the Initialize Channel (INCH) command. Valid addresses must fall within the range of the subchannel storage table (SST) locations.

Any other addresses cause the following message to be written:

```
EXP CC AND STATUS ADDRESS efffffff TO essssss
ACT CC AND STATUS ADDRESS caaaaaaa
```

CC indicates the condition code.

fffffff specifies the first valid data address as presented by the INCH command.

ssssss specifies the last valid data address of the subchannel storage table (SST).

aaaaaaa specifies the actual address received in the service interrupt receiver.

e specifies the expected condition code.

c specifies the condition code setting in the status word.

If this error message is reported, no verification is made as to the validity of the contents of the device status. Therefore, no comparisons are made between the expected and actual status received from the device.

4.7 Program Start Errors

After receiving the first Start command, the program performs a Reset Channel, Enable Channel Interrupt, and a Start I/O instruction to initialize the IOM/Controller. The following error messages are related to the first Start I/O command:

- A Reset Channel instruction failure causes the following message to be written:

```
TST:RCH,CA:cc,DSA=ss,PC:ppppp
aaaaa,DRTV:RCH
CMND:iiiiiiii
```

  cc — specifies the hexadecimal IOM/channel address assigned.

  ss — specifies the hexadecimal device subaddress assigned.

  ppppp — specifies the hexadecimal program counter value.

  aaaaa — specifies the address of the command device instruction.

  iiiiiiii — specifies the command device instruction issued.

  This message usually is accompanied by an incorrect condition code error message.

- An Enable Channel Interrupt instruction failure causes the following message to be written:

```
TST:ECI,CA:cc,DSA:ss,PC:ppppp
aaaaa,DRTV:ECI
CMND:iiiiiiii
```

  This message usually is accompanied by an incorrect condition code error message.

- An Initialize Channel instruction failure causes the following message to be written:

```
TST:ICH,CA:cc,DSA:ss,PC:ppppp
aaaaa,DRTV:ICH
CMND:iiiiiiii IOCD dddddddd dddddddd
```

  dddddddd — specifies the two-word Initialize Channel IOCD.

  This message usually is accompanied by an incorrect condition code error message, a no status interrupt received message and the following message:

```
TERMINATED: CANNOT INITIALIZE CHANNEL
```

  This message indicates that the device cannot be initialized and the program cannot proceed until the device is initialized.

- A Load Mode Register command failure causes the following message to be written:

```
TST:LMR,CA:cc,DSA:ss,PC:ppppp
aaaaa,DRTV:LMR
CMND:iiiiiiii IOCD ddddddddd ddddddddd
```

This message is accompanied by a status error and/or sense data errors. This message also is reported if errors are detected during media changing on the Cartridge Module (Model 8117).

4.8 Additional Messages

Additional messages are written to the operator communications (OC) file if an unexpected error occurs.

If the program cannot correctly perform and initialize channel command to the device, the following message is written:

```
CANNOT INITIALIZE CHANNEL
```

This message is usually accompanied by a condition code error message or a no status interrupt message. No further testing can be performed until the channel is initialized.

If no interrupt is detected but an interrupt is expected during the execution of interrupt instruction testing, the following message is written:

```
NO SI
```

This message is exclusive to execution of Test 1.

If an interrupt is detected but an interrupt is not expected during the execution of interrupt instruction testing, the following message is written:

```
UNEXPECTED SI
```

This message is exclusive to execution of Test 1.

If a reset channel directive is performed but an initialize channel command is not issued before attempting to perform another I/O directive, the following message is written:

```
CHANNEL MUST BE INITIALIZED
```

If during the execution of Test 10, a Write Channel instruction is issued to the device and the unsupported transaction condition code (CC=5) is not detected after the instruction is executed, the following message is written:

```
WCWCS - NO UNSUPPORTED TRANS STATUS
```

This message is exclusive to Test 10.

If during the execution of Test 10, a Grab Controller instruction is issued to the device and the unsupported transaction condition code (CC=5) is not detected after the instruction is executed, the following message is written:

```
GRIO - NO UNSUPPORTED TRANS STATUS
```

This message is exclusive to Test 10.

If during the execution of Test 2, no interrupt is detected after a Halt I/O instruction is executed, the following message is written:

```
IDLE STATE:  NO INTERRUPT AFTER HIO
```

If during the execution of Test 2, an interrupt is detected after a Stop I/O instruction is executed, the following message is written:

```
IDLE STATE:  INTERRUPT AFTER STP
```

If during the execution of Test 2, a Test I/O instruction does not indicate that the Test I/O instruction is accepted with no echo response, but rather gives the response as represented by an asterisk (*), the following message is written:

```
IDLE STATE:  EXPECTED CC 8
IDLE STATE:  ACTUAL CC *
```

* specifies the condition code received.

If after a Sense command is executed, the expected drive attributes are not what is expected from the drive, the following message is written:

```
ATTRIBUTE ERROR EXP=eeeeeeee, ACT=aaaaaaaa
```

eeeeeeee specifies the drive attributes expected from the disc drive.

aaaaaaaa specifies the actual attributes received from the disc drive.

This error occurs:

- If the drive is formatted as dual ported and the program is reset to single ported and an attempt is made to run the diagnostic in the single port mode without reformatting the disc or
- If a Cartridge Module Disc Drive number is assigned to an 80 MB or 300 MB Disc Drive.

4.9 Directive Error Reporting

One of the following messages is written to the operator communications (OC) file if an operator-composed directive violates a extended I/O protocol:

If more than 16 I/O command doublewords (IOCDs) are written into the I/O command list (IOCL) queue, the following message is written:

```
NUMBER OF DIRECTIVES PROCESSED WOULD HAVE CAUSED THE IOCD LIST TO OVERFLOW
```

The maximum number of IOCDs in the IOCL queue is 16.

If an attempt is made to change function during a data chain the following message is written:

CANNOT DATA CHAIN TO A DIFFERENT TYPE OF I/O COMMAND

If the last I/O command doubleword (IOCD) entered in the I/O command list (IOCL) queue has its data chain flag set and no IOCD follows (incomplete chain) the following message is written:

DIRECTIVE STRING ENDED WITH DATA CHAIN SET

If the data-chained byte count exceeds the maximum size of the output buffer the following message is written:

DATA CHAIN WOULD HAVE CAUSED OVERRUN OF THE OUTPUT BUFFER

If the program detects a Transfer-in-Channel (TIC) to a Transfer-in-Channel (TIC) command the following message is written:

'TIC' TO 'TIC' IS INVALID

This sequence causes an IOCD validation error.

If a Transfer-in-Channel (TIC) command is the first IOCD entered in the IOCL queue the following message is written:

'TIC' AS FIRST IOCD INVALID

If a Transfer-in-Channel IOCD (TIC) points forward to an IOCL queue slot into which no valid IOCD exits the following message is written:

TIC POINTS TO EMPTY IOCL QUEUE SLOT

If a Reset Channel command executes and no Initialize Channel command is entered the following message is written:

CHANNEL MUST BE INITIALIZED

If an attempt to data chain to an illegal command is made the following message is written:

CANNOT DATA CHAIN THIS COMMAND

If no entries are currently in the IOCL queue, the following message is written:

IOCL QUEUE EMPTY

If a dual port directive is issued to a subaddress which has a Cartridge Module (Model 8117) assigned to the device subaddress, the following message is written:

CANNOT DUAL PORT A CARTRIDGE MODULE DRIVE

If a Cartridge Module directive or directive test is attempted but the program is operating on a Disc Processor (Model 8055) channel, the following message is written:

PHASE 1 DOES NOT SUPPORT MODEL #8117

The Cartridge Module Drive (Model 8117) is supported only by the Disc Processor (Model 8055).

If media formatting is prematurely terminated by a program abort, Halt I/O, Stop I/O or Reset channel instruction, the following message is written:

SUBADDRESS * WAS TERMINATED DURING MEDIA FORMAT AT ????--??--??

* specifies the assigned subaddress.

? specifies the disc coordinates where termination took place.

If the first word of the bad track sector does not have F000 in the first half word of data which is input from the media, the following message is written:

1ST WD OF DEALLO TRK HAS INCORRECT IDENTIFIER

This indicates that the data on the sector is not valid and no list out of the bad tracks will be given. The contents of the sector may be displayed by using a Dump directive.

If track label zero, head zero does not contain a media descriptor pointer which enables the diagnostic to read in the appropriate deallocated tracks, the following message is written:

TRACK LABEL ZERO POINTS TO CYLINDER ZERO
CYLINDER ZERO IS INVALID FOR DEALLO TRACK USE

This message is a fatal error since only tests 1, 2, 3, 5, 6, 8, 10, 11, and 12 are executed without the program detecting the media descriptor information in track label zero. Preparation of track label zero is performed by the Gould RPU Media Verification program or by using the ..,ZAP,.. directive. The ..,ZAP,.. directive should only be used for initial media verification because it performs no other media verification except reading track and sector labels and deallocates only the default diagnostic cylinders, the hardware.

The deallocated track information is not written to the media if the ..,ZAP,.. directive is used.

Corrective Measures

- Media Formatted and Verified by the Gould RPU Disc Processor Diagnostic Rev C or D
  Media formatted with Rev C or D of the diagnostic program is converted to the new deallocation track format by using the Gould RPU Media Verification Program and using the ..,DNEW,.. directive. The Media program performs all conversions of previous detected bad tracks and establishes the new deallocated boundaries.
- Media Formatted but not Verified
  Media formatted with Rev C or D of the diagnostic program is converted to the new deallocation track format by using the ..,DNEW,.. directive in the Gould RPU Media Verification program.

If a Cartridge Module directive, such as, (..,CCM,..), (..,CMB,..) or (..,CMF,..) is entered via the operator's console and the program detects that a Cartridge model number assignment has not been made for the device subaddress reported, the following message is written:

```
SUBADDRESS * DOES NOT HAVE CARTRIDGE MODEL #8117
```

* specifies the current assigned subaddress.

4.10 Label Verification Messages

During the execution of Test 4, track and sector label flags are checked for settings other than Good Track or Sector if the system disc flag is OFF. If any of these conditions are detected, the following messages are written to the listed output (LO) file.

Mnemonic	Represents
aaa	Cylinder number
bb	Head number
cc	Starting sector number
ddd	Defective or alternate cylinder
ee	Defective or alternate head
ff	The total number of sectors starting at the value as shown in cc

If bit 1 of label flag byte 1 is set, making this an alternate track that points to defective track ddd--ee, the following message is written:

```
'taskname' aaa--bb--cc ALTERNATE=DEFECTIVE TRACK ddd--ee ff SECTOR(S)
```

If bit 2 of label flag byte 1 is set, the following message is written:

```
'taskname' aaa-bb--cc SPARE TRACK ff SECTOR(S)
```

If bit 3 of label flag byte 1 is set, the following message is written:

```
'taskname' aaa-bb-cc RESERVED TRACK ff SECTOR(S)
```

If bit 4 of label flag byte 1 is set making this a defective track that points to alternate track ddd-eee, the following message is written:

```
'taskname' aaa--bb--cc DEFECTIVE=ALTERNATE TRACK ddd---ee ff SECTOR(S)
```

If bit 5 of label flag byte 1 is set, the following message is written:

```
'taskname' aaa---bb-cc LAST TRACK ff SECTOR(S)
```

If bit 0 of label flag byte 2 is set, the following message is written:

```
'taskname' aaa--bb--cc WRITE LOCK SET ff SECTOR(S)
```

If bit 1 of label flag byte 2 is set, the following message is written:

```
'taskname' aaa--bb-cc WRITE PROTECT SET ff SECTOR(S)
```

4.10.1 Description of Label Flag Byte One

Bit	Track	Description
0	Good Track	Most tracks are good tracks.
1	Alternate Track	This track is the alternate for a defective track.
2	Spare Track	This is an unused track that may be used later as an alternate track.
3	Reserved Track	This is a track which can only be read or written if the subchannel is in the reserve track mode. Also, there must be a jumper present on the disc controller to allow this access.
4	Defective Track	This track defined as unusable which causes alternate track addressing to take place if the defective track is accessed.
5	Last Track	This bit may be set in combination with any of the orders. If a user specifies a byte count in an I/O command doubleword (IOCD) which extends past the end of this track, the I/O is terminated. In addition, if defective track is also set, then alternate track addressing does not take place.
6-7		Unimplemented.

4.10.2 Description of Label Flag Byte Two

Bit	Condition	Description
1	Write Protected	If this bit is set, it is not possible to write to the data portion of the sectors on this track.
0	Write Locked	If this bit is set, it is not possible to alter any of the labels on this track.

4.11 Track Deallocation Messages

If a track is deallocated, a message is written to the listed output (LO) file specifying the reason for deallocation. The standard track deallocation message format is as follows:

```
'taskname'   scyl--hh--ss    8raaaaaa   comment
             ecyl--hh-ss     9raaaaaa
```

taskname — specifies the one to eight character diagnostic name indicated in the load map.

scyl--hh--ss	specifies the starting cylinder (scyl), head (hh), and sector (ss) of the deallocated track.
ecyl-hh-ss	specifies the ending cylinder (ecyl), head (hh), and sector (ss) of the deallocated track.
8	indicates the starting address. The absolute binary address (aaaaaa) in the value beginning with eight is the starting sector address of the deallocated track or sector.
9	indicates the ending address. The absolute binary address (aaaaaa) in the value beginning with nine is the ending sector address of the deallocated track or sector.
r	specifies the track deallocation reason code. The reason code (identified 1 through C) identifies which of the 12 reasons the track is deallocated.
aaaaaa	specifies the absolute binary address. This address is either the starting (preceded by an eight) or ending (preceded by a nine) sector address of the deallocated track.

The following are track deallocation reason codes:

Reason Code	Comment
1	ECC IN TRACK LABEL
2	ECC IN SECTOR LABEL
3	UNCORRECTABLE DATA ECC
4	CORRECTABLE DATA ECC
5	NO SI RECEIVED
6	CPC OR INCR LENGTH
7	DISC FORMAT ERROR
8	DIAGNOSTIC BOOT LOADER BOUNDARIES
9	DIAGNOSTIC CYLINDER BOUNDARIES
A	MEDIA DESCRIPTOR BOUNDARIES
B	ADDED FALSE ADDRESS MARK
C	HARDWARE DEVICE TEST TRACK

Reason Code 1 error correction code (ECC) in Track Label

If bit 29 of sense word two is set after the diagnostic attempts to adjust address marks by reformatting the track and attempted to read track labels, the following message is written:

```
'taskname' scyl-hh-ss  81aaaaaa  ECC IN TRACK LABEL**ERROR**
           ecyl-hh-ss  91aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 2 ECC in Sector Label

If bit 27 of sense word two is set after the diagnostic attempts to adjust address marks by reformatting the track and attempts to Write or Read data, the following message is written:

```
'taskname' scyl-hh-ss  82aaaaaa  ECC IN SECTOR LABEL**ERROR**
           ecyl-hh-ss  92aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 3 UNCORRECTABLE DATA ECC

If after completing a Read ECC command, an error correction code (ECC) data error occurs, sense word 2 bit 31 is set, (indicating that the data error is uncorrectable), the following message is written:

```
'taskname' scyl-hh-ss  83aaaaaa  UNCORRECTABLE DATA ECC**ERROR**
           ecyl-hh-ss  93aaaaaa
```

The sector is considered to be a deallocated sector.

Reason Code 4 CORRECTABLE DATA ECC

If an error correction code (ECC) data error occurrs, the following message is written:

```
'taskname' scyl-hh-ss  84aaaaaa  CORRECTABLE DATA ECC
           ecyl-hh-ss  94aaaaaa
```

However if the error is corrected by using the Read ECC command, and by applying the displacement and correction mark to the data in error, resulting in the data being corrected, the sector is considered a potential deallocated sector.

Reason Code 5 NO STATUS INTERRUPT RECEIVED

If no interrupt is received while attempting to read Track Labels or data, the following message is written:

```
'taskname' scyl-hh-ss  85aaaaaa  NO SI RECEIVED
           ecyl-hh-ss  95aaaaaa
```

This message indicates that a false address mark may have been detected. This also indicates that the track cannot be accessed by write or read commands and is therefore a deallocated track.

Reason Code 6 CHANNEL PROGRAM CHECK OR INCORRECT LENGTH

If after writing data to the media, the program attempts to read a track label which does not have a 30-byte count record size, the following message is written:

```
'taskname' scyl-hh-ss  86aaaaaa  CPC OR INCR LENGTH**ERROR**
           ecyl-hh-ss  96aaaaaa
```

This error also occurs if the track does not have the correct sector format, i.e., 16-sector format instead of 20-sector format.

Reason Code 7 DISC FORMAT ERROR

If sense word 2 bit 14 is set after the diagnostic attempts to adjust address marks by reformatting the track and attempts to read track labels or data, the following message is written:

```
'taskname' scyl-hh-ss  87aaaaaa  DISC FORMAT ERROR**ERROR**
           ecyl-hh-ss  97aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 8 DIAGNOSTIC BOOT LOAD BOUNDARIES

The following message is established for future use:

```
'taskname' scyl-hh-ss  88aaaaaa  MEDIA DESCRIPTOR BOUNDARIES
           ecyl-hh-ss  98aaaaaa
```

Reason Code 9 DIAGNOSTIC CYLINDER BOUNDARIES

If two cylinders that are defined during Media Verification, are used by the diagnostic program during program execution with the System Disc Flag 'ON', the following message is written:

```
'taskname' scyl-hh-ss  89aaaaaa  DIAGNOSTIC CYLINDER BOUNDARIES
           ecyl-hh-ss  99aaaaaa
```

These two cylinders are deallocated and are devoted exclusively to the Gould RPU Disc Processor Diagnostic by the designated operating system.

Reason Code A Media Descriptor Boundaries

If deallocated track information is within the boundaries of this cylinder, the following message is written:

```
'taskname' scyl-hh-ss  8Aaaaaaa  MEDIA DESCRIPTOR BOUNDARIES
           ecyl-hh-ss  9Aaaaaaa
```

The Gould RPU Media Verification program establishes deallocated tracks and verifies that the sectors which contain this information are error-free.

Reason Code B Added False Address Mark

If a track is deallocated by the Gould RPU Media Verification Program ..,DT,.. directive, the following message is written:

```
'taskname' scyl-hh-ss  8Baaaaaa  ADDED FALSE ADDRESS MARK
           ecyl-hh-ss  9Baaaaaa
```

The ..,DT,.. directive allows for the insertion of bad tracks to the Deallocated Track Table. The diagnostic program does not have the capability of adding false address marks.

Reason Code C Hardware Device Test Track

If the last track on the media is deallocated to both the Diagnostic Program and to the designated operating system, the following message is written:

```
'taskname' scyl-hh-ss  8Caaaaaa  HARDWARE DEVICE TEST TRACK
           ecyl-hh-ss  9Caaaaaa
```

This track is used exclusively by the Hardware Drive Test Equipment.

The following message is written to the listed output (LO) file after format and label verification, or media verification is completed:

```
'taskname' ** TRACK SUMMARY ** CA:xx,DSA:yy
```

If any bad tracks are detected, this message is followed by a summary of bad (defective) tracks which are detected during the execution of the test.

If a maximum of 280 errors have been detected on the media being tested, the following message is written:

```
'taskname' DEALLOCATION TABLE IS FULL
```

The bad track data information is written to the designated track and Test 4 is terminated.

CHAPTER 5

DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Compiler

The Gould RPU Disc Processor Diagnostic (RP.DPD.r) is structured around the capabilities of the Directive Language Compiler. The Directive Language Compiler (a component of the Diagnostic Executive Program), and its associated run-time package (a component of the diagnostic) use directive language to offer the following diagnostic capabilities:

- The ability to reduce a diagnostic test into its failing components for troubleshooting.
- The ability to create new directive test sequences.

The basic component of the directive language is the directive. For all Level Two diagnostics, there are two categories of directives -- standard and unique. The DEXP provides standard directives for all Level Two diagnostics, while each Level Two diagnostic contains unique directives to perform specific functions within the individual program.

A complete understanding of standard and unique directives is needed to use the directive language effectively.

5.2 DEXP Standard Directives

The Diagnostic Executive Program contains nine standard directive categories. The following is a list of standard directives and their functions:

5.2.1 "A" Directives (Assign)

Three "A" directives assign channel, controller, and unit addresses for each controller.

..,A,..

Uses the channel address and the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,AC,..
..,ACn,..
..,ACA,..
..,ACAn,..

Uses the specified channel address n or, if n is not specified, the channel address currently in the device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,ADSA,..
..,ADSAn,..
..,AU,..
..,AUn,..

Uses the specified device subaddress n, or if n is not specified, the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

5.2.2 "C" Directives (Cycle)

Four "C" directives cycle the output buffer (equivalent to a circular shift with end-to-end wraparound).

..,CL,..
..,CLn,..

Cycles the output buffer left by number n byte positions. If number n is not specified, the output buffer is cycled left by one byte position.

..,CR,..
..,CRn,..

Cycles the output buffer right by number n byte positions. If number n is not specified, the output buffer is cycled right by one byte position.

..,CLp,..

Cycles the output buffer left by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled left by one byte position.

..,CRp,..

Cycles the output buffer right by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled right by one byte position.

5.2.3 "D" Directives (Dump)

Four "D" directives dump all or any specified part of the diagnostic's input buffer, output buffer, or parameter set to the listed output (LO) file. If the LO file is assigned to a line printer, the byte count is rounded up to the next multiple of 32 bytes (eight words). Otherwise, the byte count is rounded up to the next multiple of 16 bytes (four words).

..,DI,..
..,DIn,..

Dumps number n or, if number n is not specified, parameter W bytes of the diagnostic's input buffer. If number n is not specified and parameter W equals zero, the entire input buffer is dumped.

..,DO,..
..,DOn,..

Dumps number n or if number n is not specified, parameter W bytes of the diagnostic's output buffer. If number n is not specified and parameter W equals zero, the entire output buffer is dumped.

..,DP,..

Dumps the contents of all parameters (A through Z).

..,DPp,..

Dumps only the contents of parameter p (any specified parameter A through Z).

5.2.4 "H" Directives (Hold)

Two "H" directives hold diagnostic execution. These directives establish delays between directives.

..,H,..
..,Hn,..

Holds diagnostic execution for the specified number of time units n or, if number n is not specified, for the number of time units in parameter X. If parameter X contains zero, the hold duration is one time unit (determined by the Real-Time Clock rate). If the Real-Time Clock rate is 120 Hz, one time unit equals 8.3 ms.

..,HS,..
..,HSn,..

Holds diagnostic execution for the specified number of seconds n or if number n is not specified, for the number of seconds in parameter X. If parameter X contains zero, the hold duration is 1 second.

5.2.5 "J" Directives (Jump)

Five "J" directives transfer control in a directive string. The following are the relational operators to use with conditional jumps:

(op):		
	-	not
	=	equal to
	>	greater than
	<	less than
	=>	equal to or greater than
	>=	greater than or equal to
	<=	less than or equal to
	=<	equal to or less than

..,Jn,..

Jumps indefinitely to statement number n (0<=n<=9).

..,JZ,..

Jumps indefinitely to the statement number in parameter Z.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k,..

Jumps to statement number n or the statement number in parameter Z, k times (1<k<=32,767).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p,..

Jumps to statement number n or the statement number in parameter Z the number of times defined by the contents of the specified parameter (A through Z).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k,..

Jumps to the statement number n or the statement number in parameter Z as long as the contents of p (any specified parameter A through Y) satisfies the stated relationship (op) with the constant k.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2,..

Jumps to statement number n or the statement number in parameter Z as long as the contents of p_1 (the first specified parameter A through Z) satisfies the stated relationship (op) with the contents of p_2 (the second specified parameter A through Y).

5.2.6 "L" Directives (Load)

Fifteen "L" directives load input or output buffers with a variety of data test patterns.

..,L=' ',..

Loads the output buffer with 'any literal value' and stores the literal string length in parameter W.

..,LA,..
..,LAn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the 64-character printable ASCII subset. If parameter W equals zero, the entire output buffer is loaded.

..,LIZ,..
..,LIZn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with all zeros. If parameter W equals zero, the entire input buffer is loaded.

..,LZ,..
..,LZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all zeros. If parameter W equals zero, the entire output buffer is loaded.

..,LO,..
..,LOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all ones. If parameter W equals zero, the entire output buffer is loaded.

..,LFZ,..
..,LFZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating zero (FE,FD,FB,...,BF,7F,FE,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LFO,..
..,LFOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating one (01,02,04,...,40,80,01,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LBK,..
..,LBKn,..

Loads number n or, if number n is not specified, parameter W bytes to the output buffer with ASCII blanks (20,20,20,...,20,20,20,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBP,..
..,LBPn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary progression (00,01,02,...,FE,FF,00,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBR,..
..,LBRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary regression (FF,FE,FD,...,01,00,FF,...). If parameter W equals zero, the entire output buffer is loaded.

..,LIR,..
..,LIRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the initial random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LLR,..
..,LLRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the last random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LNR,..
..,LNRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the next random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LOI,..
..,LOIn,..

Loads number n or if number n is not specified, parameter W bytes of the output buffer with the contents of the input buffer. If parameter W equals zero, the entire output buffer is loaded.

..,LIO,..
..,LIOn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with the contents of the input buffer. If parameter W equals zero, the entire output buffer is loaded.

5.2.7 "P" Directives (Parameter)

Five "P" directives manipulate entries in the diagnostic's parameter table. The following are arithmetic and logical operators to use with "P" directives:

(op):		
	=	replacement
	+	addition
	-	subtraction
	*	multiplication
	/	division
	.A.	logical AND
	.O.	logical OR
	.E.	logical EXCLUSIVE OR
	.N.	two's complement
	.C.	one's complement
	>	circular shift right
	<	circular shift left

..,Pp(op)k,..

The contents of the specified parameter (p specifies a parameter A through Z) is fetched then the logical OR arithmetic operation is performed. The result then is stored into the specified parameter the (range of k is zero through 32,767).

..,Pp$_1$(op)p$_2$,..

The contents of the specified parameters p_1 and p_2 are fetched, then the logical OR arithmetic operation is performed. The result then is stored in the specified parameter p_1.

..,PD(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of parameter D (and the second parameter if also specified) are fetched, then the logical OR arithmetic operation is performed. The result then is stored in parameter D and also into the entire output buffer.

..,PI(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of parameter I and either the second parameter Pp (if specified) or k are fetched then the logical OR arithmetic operation is performed. The result is stored into parameter I and also into word one of the output buffer.

..,PZ= $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of Pp (P specifies any parameter A through Y) or k, is stored into parameter Z. The value of parameter Z represents a statement number to which the ..,JZ,.. directive transfers control. The only valid operator for this directive is = (replacement). The contents of the parameter specified must be between zero and nine.

5.2.8 "S" Directive (Stop)

..,STPn,..

The "S" directive creates convenient breaks in a directive string for performing necessary functions.

Stops diagnostic execution and writes the following message to the operator's console:

```
'TASKNAME' ,STPn, DIRECTIVE ; [C,H,T] ?
```

5.2.9 "T" Directive (Time)

The "T" directive writes the time-of-day and an informative message (if specified in the directive) to the listed output (LO) file.

..,TIME,..
..,TIME (message),..

5.3 Gould RPU Disc Processor Unique Directives

The Gould RPU Disc Processor Diagnostic contains unique directives to perform specific functions within the diagnostic. Many unique directives, however, cross diagnostic boundaries and are used in different diagnostics.

The following is a list of the Gould RPU Disc Processor unique directives:

5.3.1 "A" Directives

..,ADP,..

Assigns the correct set of disc drive parameters for each specified drive in an ..,AU,.. (assign unit) directive string. These parameters consist of the media size, beginning, and ending addresses for diagnostic testing.

Note: This directive should always immediately follow the ..,ADSA,.. directive in a string to insure that the correct parameters are selected each time a different model drive is assigned.

..,AMCn,..

Sets the first address mark n bytes from the index pulse in the track format address mark data which is used by the Format Disc command. The directive should be used prior to a Format Track directive ..,FT=,.. to adjust address marks after a track has been found to have a media flow. The address mark adjustment remains in the track format address mark data until the ..,AMO,.. directive is used to reset the first address mark to the original data value of 21 bytes.

Consider the following:

- Program has the name D1
- Drive is connected to subaddress six
- Cylinder 100, head two has been found to have a media flaw
- Address mark adjustment is 70

```
DXP>D1 AU6,AMC70,FT='THC(100,2)'
```

This command should be followed by resetting the address marks back to the original data:

```
DXP>D1 AU6,AMO
```

..,AMO,..

Sets the first address mark in the track format address mark data to 21 bytes from the index pulse (the default condition).

..,ANG,..
..,ANG='arg_1,arg_2,...,arg_n',..

Inserts a Read Angular Position (ANG) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed. For a description of the optional subfields, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,AUn,..

Assigns the device subaddress specified by n as the device subaddress of the disc drive to test, and aligns the assigned device subaddress with the correct number parameters for the specified drive. This directive is not needed if only one disc drive is configured.

Note: This directive should be immediately followed by the ..,ADP,.. (assign drive parameters) directive to insure that the correct parameter set is selected for the drive being assigned.

5.3.2 "B" Directives

..,BFE,..

Relocates the diagnostic's input buffer to the starting address of the output buffer. This directive is used to input data from one drive and have the data positioned for output to another drive. The ..,BFI='PGM',.. and ..,BFO='PGM',.. directives reset the starting addresses of the input and output buffers, respectively.

..,BFI='address',..

Relocates the diagnostic's input buffer to the specified address.

A difference of 3000 should be allowed between the input and output buffers to insure that maximum byte count transfers do not cause buffer overflows.

..,BFI='PGM',..

Relocates the diagnostic's input buffer to the default location in the diagnostic program.

..,BFO='address',..

Relocates the diagnostic's output buffer to the specified address.

..,BFO='PGM',..

Relocates the diagnostic's output buffer to the default location in the diagnostic program.

5.3.3 "C" Directives

..,CCM,..

Sets parameter Y to the number of fixed heads in the Drive Attribute Table.

..,CDP,..

Sets parameter B to the value of parameter K if the drive is not dual ported or if the System Disc Flag is OFF. If the drive is dual ported or the System Disc Flag is ON, parameter B has the value of parameter K-1. If either of the last two conditions are true, the value in parameter B prevents Tests 21 and 22 from performing head advance over the designated diagnostic cylinders.

..,CEC,..

Fetches the error correction data word from a Read Error Correction Code (ECC) command. The byte displacement is calculated from the bit displacement of the ECC word and the mask is Exclusive Orred into the input buffer. This directive is followed by the ..,COM,.. directive to insure that the data is correct.

..,CMF,..

Conditions a flag which indicates to the diagnostic that disc drive access may be made only to the captive (fixed) disc on the cartridge module drive. Once this directive is entered, the diagnostic tests and accesses only the captive disc portion of the drive. The disc coordinates accessed by the diagnostic are dependent upon the System Disc Flag setting for this subaddress. If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive.

..,CML,..

Sets a flag in the Device Parameter Block for the currently assigned unit if the following criteria has been established:

(1) Track Label Zero is input and a pointer is found in bytes 12 through 15 which is used to access the deallocation track information in those sectors.

(2) After reading the Deallocation Track Table, an FO is found in word zero indicating a start of the deallocated information.

(3) The diagnostic is able to find an 89 and 99 entry in the table, which is used to establish the boundaries for diagnostic cylinder testing and the boundaries for testing the entire media if the System Disc Flag is OFF.

(4) If none of the above criteria are established and if track label zero may be accessed properly, the diagnostic defaults to the model number drive parameters.

This directive establishes a worst case test pattern of 5A5A5A5A for all drives except model numbers 8136 and 8146 which use a worst case test pattern of C67BC67B.

The diagnostic program then is capable of executing all tests within the constraints of the System Disc Flag setting.

Failure to properly perform any of the above criteria results in restricting the diagnostic program to performing Tests 1, 2, 3, 5, 6, 7, 8, 10, 11, and 12. No other tests are performed.

..,CMM,..

Conditions a flag which indicates to the diagnostic that disc drive access may be made only to the removable disc pack on the cartridge module drive. This directive causes the diagnostic to test and access only the disc pack on the drive. The disc coordinates accessed by the diagnostic are dependent upon the System Disc Flag setting for this subaddress. If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive.

..,CNO,..

Clears parameter Y. Fetches the fixed head number if it is a fixed head cartridge module and stores it in parameter Y.

..,COM,..

Compares the contents of the input buffer with the contents of the output buffer and reports any detected errors.

..,CONn,..

Conditions a flag which indicates to the diagnostic program which type of controller is being used by the drive. The diagnostic program automatically defaults to controller model number 8055. This directive allows the diagnostic to function with either controller, Model 8055 or 8060. If more than one drive has been assigned to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive.

n specifies:

8055	Phase 2 Controller model number
8060	Universal Disc Processor

..,CPS,..

Clears parameter Y. Sets bit in parameter Y if it is a phase two controller for Test 40 execution.

..,CPT,..

Clears parmeter Y. Increments parameter Y if dual 1 or dual 2 is set for Test 25 execution.

..,CSF,..

Selects fixed head from the attributes and sets the flag for fixed head disc access only.

..,CSM,..

Sets the fixed head count to parameters F and K.

..,CVF,..

Uses the value in parameter C to calculate the next head coordinate being used by the fixed portion of the cartridge module. This directive also establishes the current values for parameters F and K by using the number of fixed heads obtained from the drive attribute register.

..,CVM,..

Uses the value in parameter B to calculate the next head coordinate being used by the removable portion of the cartridge module. This directive also establishes the current values for parameters F and K by using the number of moving heads obtained from the drive attribute register.

..,CVX,..

Sets register 2 to the disc flag word of the currently assigned unit.

..,CWL,..

Clears parameter H. Fetches the starting cylinder in parameter L and stores it in parameter T for use in Tests 27 and 28. If the system disc flag is off and it is a cartridge module, it increments parameter T. If it is not a cartridge module, it positions the head to start past head zero.

5.3.4 "D" Directives

..,DAL0,..
..,DUAL0..,

Dual Port Using Entire Disc

Conditions parameters F, G, K, L, O, Q, and U to the original values indicated by the selected model number. The diagnostic then performs Reset Channel and Enable Channel interrupt instructions followed by an Initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the dual port mode. (Refer to Disc Drive Default and Test Parameters.)

If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive. This directive should be used to format the media prior to dual port testing. This directive should always be preceded by the System Disc Flag directive to insure that the disc drive test parameters are set properly.

..,DAL1,..
..,DUAL1,..

Dual Port Using Lower Half of Disc Pack

Conditions parameters F, G, K, L, O, Q, and U to the first half of the valid tracks indicated by the selected model number. The diagnostic then performs Reset Channel and Enable Channel interrupt instructions, followed by an Initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the dual port mode. The diagnostic utilizes the first half of the disc pack or the first diagnostic track. (Refer to Disc Drive Default and Test Parameters.)

If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive. This directive should always be preceded by the System Disc Flag directive to insure that the disc drive test parameters are set properly.

..,DAL2,..
..,DUAL2,..

Dual Port Using Upper Half of Disc Pack

Conditions parameters F, G, K, L, O, Q and U to the second half of the valid tracks indicated by the selected model number. The diagnostic then performs a Reset Channel and Enable Channel interrrupt instruction, followed by an initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the dual ported mode. The diagnostic utilizes the second half of the disc pack or the second diagnostic track. (Refer to Disc Drive Default and Test Parameters.)

If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive. This directive should always be preceded by the System Disc Flag directive to insure that the disc drive test parameters are set properly.

..,DAT,..

Writes the contents of all Drive Attribute Registers to the listed output (LO) file.

Drive Attribute Register layout is as follows:

Bits 0 — 7	8 — 15	16 — 23	24 — 31
Flags	Sector Count	MHD Count	FHD Count

Bits	Meaning
0	10 = FHD (Fixed Head Disc)
1	01 = MHD (Moving Head Disc) 11 = MHD with FHD option 00 = Reserved
2	1 = Cartridge Module Drive
3	Reserved
4	1 = Drive Not Present
5	1 = Drive is Dual Ported
6-7	Reserved for future expansion, must be zero

Sector Count — specifies the number of sectors per track (14_{16}).

MHD Count — indicates the number of heads on the MHD, or the number of heads for the removable disc pack portion of the cartridge module drive.

FHD Count — indicates the number of heads for the captive disc portion of the cartridge module drive.

The Drive Attribute Register model number 9342 (80 megabyte) dual ported, moving head disc, is generated as follows:

DATAW 44140500$_{16}$

- Zero FHD count
- Five MHD count
- 20 Sectors per Track
- Dual port indication
- Moving Head Disc

The Drive Attribute Register for a model number 8104 (5 megabyte) single port, fixed media disc, is generated as follows:

DATAW 40140400_{16}

- Zero FHD count
- Four MHD count
- 20 Sectors per Track
- Moving Head Disc

The Drive Attribute Register for a model number 8117 (32 megabyte) cartridge module drive is generated as follows:

DATAW 20140101_{16}

- One FHD count
- One MHD count
- 20 Sectors per Track
- Cartridge Module Drive

The Drive Attribute Register for a drive that is not present is generated as follows:

DATAW 0800000_{16}

- Drive Not Present

..,DBT,..

Dumps the contents of the deallocated (bad) track table to the listed output (LO) file.

Note: This directive should be preceded by both the ..,AU,.. (assign unit) and the ..,ADP,.. (assign drive parameters) directives.

..,DDP,..

Dumps the drive parameter table (parameters A through Z).

..,DEC,..
..,DEC='arg_1,arg_2,..,arg_n',..

Inserts address of disabled Error Correction Code (ECC) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. This command may be command chained to a read command and prevents the cache disc accelerator from automatically correcting data if an ECC is present. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed.

The legal argument for this directive is: CC

..,DMPn,..

Dumps the current sector to the listed output (LO) file and continues dumping sectors until number n sectors are dumped, or the end of the media is reached. If no number is specified, the entire disc from the current location to end of media will be dumped.

..,DQ,..

Dumps the contents of the I/O command list (IOCL) queue to the LO file.

..,DSD,..

Dumps the sense data received from the last executed Sense command to the LO file.

..,DST,..

Dumps the channel/device status received from the last executed Start I/O (SIO) command to the LO file.

5.3.5 "F" Directives

..,FDK,..
..,FDK='arg_1,arg_2,...,arg_n',..

Inserts a Format Disc (FDK) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,FORn,..

Selects either a 16- or 20-sector track format for the assigned device. The permissible values for n are 20 (for a 20-sector track format), and 16 (for a 16 sector track format). If this directive is used, the following steps are performed by the diagnostic.

1. Sets up the drive attributes to the 16- or 20-sector data values specified.
2. Executes a Reset Channel instruction.
3. Executes an Enable Channel interrupt instruction.
4. Initializes the channel to the new device attribute.

Test 4 should be performed to insure that the disc drive is properly formatted for further testing. The formats are as follows:

Sector Format Type	Bytes/Sector
n=16	1024
20	768

..,FT='THC(a,b)',..

Automatically seeks to track address a and head address b and reformats the specified track. This directive is used to format a defective track.

This directive always terminates the directive string.

5.3.6 "H" Directives

..,HELP,..

Writes the following messages to the LO file:

```
THE SYSTEM DISC FLAG IS SET OR RESET BY THE
SDF DIRECTIVE, SDF = 'SYS(X)' WHERE X=ON OR OFF
THE VALUE OF THE SYSTEM DISC FLAG MAY BE
DISPLAYED BY THE TDF DIRECTIVE

TEST 1, 2, 3, 5, 7, 8, 10, 11, 12 MAY BE USED FOR INITIAL DRIVE
CHECKOUT. THESE TESTS WILL ALSO RUN ON UNFORMATTED MEDIA
```

..,HIO,..

Issues a Command Device (CD) Halt I/O instruction.

5.3.7 "I" Directives

..,IHA,..
..,IHA='arg_1,arg_2,...,arg_n',..

Inserts an Increment Head Address (IHA) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,INCH,..
..,INCH= ' arg_1,arg_2,...,arg_n',..

Inserts an Initialize Channel (INCH) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

5.3.8 "L" Directives

..,LAD,..
..,LADn,..

Loads n number of bytes, or if n is not specified, parameter W bytes of the diagnostic's output buffer with the track, head, and sector address coordinate values. If parameter W equals zero, the entire buffer is loaded.

..,LID,..
..,LIDn,..

Loads the current identifier track, head, and sector address coordinates, TRKHDSEC, into the first word of the diagnostic's output buffer, then if n is specified, calculates and inserts n number of successive identifier values into the output buffer. Words two through 192 of the output buffer remain unchanged.

..,LLS='arg_1,arg_2,...,arg_n',..

Creates an output buffer of sector labels which are modified by specified arguments. For a description of the optional specifications, refer to Track and Sector Flag Byte Arguments.

..,LLT,..
..,LLT='arg_1,arg_2,...,arg_n',..

Creates an output buffer of one track label which is modified by specified arguments. The default condition of this directive is to set the track label to Good in the Track Flag byte. For a description of the optional specifications, refer to Track and Sector Flag Byte Arguments.

Rules

1. The sequence for preparing the output buffers is sector labels first, followed by the track label.
2. The diagnostic uses the current contents of the diagnostic disc coordinate to build sequential sector addresses.
3. The ..,LLS,.. and ..,LLT,.. directives are used to prepare an output buffer prior to the read track or read sector labels to verify that the labels are correct if the diagnostic performs a data compare.

..,LMR,..
..,LMR=',arg_1,arg_2,...,arg_n',..

Inserts a Load Mode Register (LMR) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default condition of this directive is to clear the mode register to zero. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to Load Mode Register Arguments.

..,LOOP,..

Sets the string address of this directive to the error return address. This allows keyboard Loop on Errors from this point in the directive string if option 3 (repeats sequence detecting an error) is set and an error is detected.

..,LPL,..
..,LPL='arg_1,arg_2,...,arg_n',..

Inserts a Lock Protected Label (LPL) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,LRA,..

Uses parameter B as an index into the output buffer and selects a random number. If not dual ported or if a cartridge module and the system disc flag is OFF, it uses parameter O to calculate the track address. If dual ported or not a cartridge module, parameter L is used as the starting parameter to calculate the track address. The track head sector coordinates are inserted into word 1 of the output buffer.

5.3.9 "M" Directive

..,MODn,..

Initializes the disc drive and test parameters of the current run-time subaddress to the default condition of the system disc flag ON and single port mode of operation. Refer to Disc Drive and Test Parameters. If more than one type of disc drive is to be tested, the run-time assign directive ..,AU,.. with the prescribed subaddress should precede the model number directive. This enables the diagnostic to set up the proper disc drive parameters for up to eight drives (device subaddresses) with any mixture of model numbers.

The following are model numbers:

Model #	Cylinder	Heads	Size		Description
8104	64	4	5 MB		Fixed Head
8117	823	2	32 MB	14-in.	Cartridge
8122	823	5	80 MB	14-in.	Winchester
8127	823	10	160 MB	14-in.	Winchester
8136	823	5	80 MB	9-in.	Cartridge
8146	823	10	160 MB	9-in.	Winchester
8155	843	40	675 MB	14-in.	Winchester
8420	823	5	80 MB	8-in.	Winchester
8424	1024	8	167 MB	8-in.	Winchester
8858	711	24	340 MB	9-in.	Winchester
9342	823	5	80 MB	14-in.	Removable
9344	823	19	300 MB	14-in.	Removable

Other directives which follow the model number directive to set appropriate parameters for specific testing are the ..,CMF,.. and ..,CMM,.. directives for the Cartridge Module and the ..,CON,.. directive which allows the diagnostic to test Disc Processor Channel/Controller Model 8055.

Note: The System Disc Flag is always defaulted to ON if this directive is processed.

Consider the following:

- An 80 MB disc drive is connected to device subaddress two.
- Device subaddress two is to be dual port.
- Device subaddress two is to have the System Disc Flag OFF.
- A Cartridge Module Drive is connected to subaddress four.
- Device subaddress four is to have the System Disc Flag OFF.
- Only the removable disc pack on the Cartridge Module is to be tested.
- The program was loaded under the name D1.
- The Disc Processor Channel address is 08.

The procedure would be as follows:

1. ASSIGN D1,DM08,2,4
2. Return control to the Gould Diagnostic Executive Program (DEXP) by entering @@A on the IOP console.
3. DXP>D1,AU2,MOD9342,SDF='SYS(OFF)', DUAL0(cr)
4. The WHAT NEXT ? prompt is written.
5. Return control to the DEXP by entering @@A on the IOP console.
6. DXP>D1,AU4,MOD8117,SDF='SYS(OFF)',CMM(cr)
7. Start the test.

5.3.10 "N" Directive

..,NOP,..
..,NOP='arg_1,arg_2,...,arg_n',..

Inserts a No Operation (NOP) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

5.3.11 "R" Directives

..,Rn,..

Inserts a Read I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by n, or if n is not specified, uses the value specified in parameter W as the byte count.

All Read commands use the current track, head and sector address coordinates in the sector target address register (STAR).

..,RCH,..

Issues a Command Device (CD) Reset Channel instruction.

..,RCT,..

Issues a Command Device (CD) Reset Controller instruction.

..,RDX,..
..,RDX='arg_1,arg_2,..,arg_n',..

Inserts a Read sector label and ECC I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. This directive reads in one sector of data and the ECC data associated with the sector data. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

This directive is valid for the cache disc accelerator only.

..,REC,..
..,REC='arg_1,arg_2,..,arg_n',..

Inserts a Read ECC I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed. The legal argument for this directive is: CC

This directive is valid for the cache disc accelerator only.

..,REL,..
..,REL='arg_1,arg_2,...,arg_n',..

Inserts a Release I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,REV,..
..,REV='arg_1,arg_2,...,arg_n',..

Inserts a Reserve I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,RLS,...
..,RLS='arg_1,arg_2,...,arg_n',..

Inserts a Read Sector Label I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,RLT,..
..,RLT='arg_1,arg_2,...,arg_n',..

Inserts a Read Track Label I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,RRn,...
..,RR='arg_1,arg_2,...,arg_n',..

Inserts a Read I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads n number of records of parameter W bytes, or if n is not specified, the diagnostic reads the number of records specified by parameter R. If parameter W is less than four bytes, the diagnostic defaults to four bytes. If command chain (CC) or data chain (DC) is specified, the appropriate bit in IOCD word 2 is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, DC, SKIP, SLI, NSIO.

..,RRF,..
..,RRF='arg_1,arg_2,..,arg_n',..

Inserts a Read sector and ECC I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC.

This directive is valid for the cache disc accelerator only.

..,RRM,..
..,RRM='arg_1,arg_2,...,arg_n',..

Inserts a Reset Reserved Track Mode I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

5.3.12 "S" Directives

..,SDF='SYS $\left\{\begin{matrix}(ON)\\(OFF)\end{matrix}\right\}$',..

Setting the system disc flag to OFF enables the diagnostic to access all tracks and heads on the selected model number drive. This directive should always precede the dual port directives ..,DUAL0,.. ..,DUAL1,.. ..,DUAL2,.. and ..,SING,.. the Single Port directive. This enables the diagnostic to write/read the entire media. The starting and ending disc coordinates are dependent on the system disc flag setting and whether or not dual port is being used. (Refer to Disc Drive and Test Parameters.) Each drive has its own system disc flag setting, to allow selective testing of the media. Setting the system disc flag to OFF should be used with caution and should be used only if the entire media is to be tested.

If more than one type of drive is being tested, the system disc flag directive should be preceded by the run-time assign directive ..,AU,.. with the correct device subaddress for that drive.

Setting the System Disc Flag ON enables the diagnostic to write only on the designated diagnostic cylinders as determined by the model number of the drive or by the cylinders obtained in the Deallocation Track Table. This enables the diagnostic to read labels, random seek and read labels, and to write and read data and labels on the diagnostic tracks. Data is not written to any tracks other than the designated diagnostic track(s). This capability allows a drive to be tested without destroying the integrity of the labels or data on the media other than the diagnostic track(s).

The program defaults to the system disc flag setting of ON. For the device parameters used by the diagnostic, refer to the Disc Drive and Test Parameters.

..,SDP,..

Sets the dual port drive attributes.

..,SEK,..
..,SEK='arg_1,arg_2,..,arg_n',..

Inserts a Seek I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The track coordinate is obtained from the current contents of TRKHDSEC and is placed into the disc coordinate table with the data address of the Seek command pointing to the coordinate in the table. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,SHn,..

Sets the head address coordinate n into parameter H which is used for subsequent Seek commands (where 0=<(n)=<K).

..,SIN,..

Issues a Command Device (CD) Start I/O instruction without waiting for an interrupt.

..,SING,..
..,SNG,..

Conditions parameters F, G, K, L, O, and U to the original values indicated by the selected model number. The diagnostic then performs a Reset Channel instruction and an Enable Channel Interrupt instruction followed by an Initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the single port mode (bit 5 reset in the drive attribute table). (Refer to Disc Drive Test Parameters.)

..,SIO,..

Issues a Command Device (CD) Start I/O instruction.

..,SNS,...
..,SNS='arg_1,arg_2,...,arg_n',..

Inserts a Sense I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,SPH,..

Sets the head address coordinate contained in parameter H into TRKHDSEC for subsequent Seek commands (where 0=<H=<K).

..,SPS,..

Sets the sector address coordinate in parameter S into TRKHDSEC (where 0=<S<=F).

..,SPO,..
..,SPO='arg_1,arg_2,...,arg_n',..

Inserts a Priority Override I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to the IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,SRM,..
..,SRM='arg_1,arg_2,...,arg_n',..

Inserts a Select Reserved Track Mode I/O command double (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to the IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,SSn,..

Selects the sector address coordinate specified by n for subsequent Read/Write instructions (where 0<=n<J).

..,SSP,..

Issues a Command Device (CD) Start I/O instruction without waiting for an interrupt.

..,STB,..

Fetches the current contents of parameter B. Calculates the track address from parameter G and updates the disc coordinates for subsequent Seek commands.

..,STn..,

Sets the track coordinate n to a dummy parameter which is used to test sector and track label defective/alternate, write/read commands (where 0<=n<=U).

..,STP,..

Issues a Command Device (CD) Stop I/O instruction.

..,STZ,..
..,STZ='arg_1,arg_2,...,arg_n',..

Inserts a Rezero I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

5.3.13 "T" Directives

..,TDA='DEF (a,b), ALT (x,y)'

a	specifies the defective track.
b	specifies the defective head.
x	specifies the alternate track.
y	specifies the alternate head.

Automatically seeks to track address a and head address b and conditions the track and sector flag byte labels to indicate that the track is a defective track (bit 4 set). The alternate track address x and the alternate head address y are inserted into the appropriate fields in all the labels. The program then seeks to track x and head y and conditions the track and sector flag bytes labels to indicate that the track is an alternate track (bit 1 set). In addition, the defective track address a and the defective head address b are inserted into the appropriate fields in all the labels.

This directive always terminates the directive string. Therefore, no other directive should follow this directive as part of a string of directives.

..,TDF,..

Dumps the system disc flag setting of the currently active unit to the listed output (LO) file. The message format is as follows:

'taskname' SYSTEM DISC FLAG CAxx,DSA:y IS { ON / OFF }

xx	specifies the controller address.
y	specifies the device subaddress.

..,TES,..
..,TES='arg_1,arg_2,...,arg_n',..

Inserts a Test Sector Target Address Register (STAR) I/O command doubleword (IOCD) into the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,TGOOD='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to Good track (bit 0 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a	specifies the track.
b	specifies the head.

..,TICn,..
..,TIC='arg_1,arg_2,..,arg_n',..

Inserts a Transfer-In-Channel (TIC) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If the TIC is executed, control is transferred to the IOCD in the IOCL queue slot number specified by n. A TIC is an implied command chain (CC) operation as the flag byte of the previous IOCD is passed through the TIC. Therefore, a TIC is never the first IOCD in the IOCL queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

..,TIO,..

Issues a Command Device (CD) Test I/O instruction.

..,TLAST='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to last track (bit 5 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a	specifies the track coordinate.
b	specifies the head coordinate.

..,TLOK='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to the write lock (bit zero of Label Flag byte two) mode.

This directive always terminates the directive string.

a	specifies the track coordinate.
b	specifies the head coordinate.

..,TPRO='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to the write protect mode.

This directive always terminates the directive string.

a	specifies the track coordinate.
b	specifies the head coordinate.

..,TREV='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to reserved track (bit 3 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a	specifies the track coordinate.
b	specifies the head coordinate.

..,TSP,..
..,TSP='arg_1,arg_2,...,arg_n',..

Inserts a Test Sector Target Address Register (STAR) I/O command doubleword (IOCD) into the I/O command list (IOCL) queue and sets appropriate flags. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

This directive is valid for the cache disc accelerator only.

..,TSPAR='THC (a,b)',..

Automatically seeks to track address a and head address b and sets the flag byte labels in the track and sector labels to spare track (bit 2 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a	specifies the track coordinate.
b	specifies the head coordinate.

5.3.14 "W" Directives

..,W,..
..,Wn,..
..,W='arg_1,arg_2,...,arg_n',..

Inserts a Write I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic writes the number of bytes specified by n, or if n is not specified, uses the value specified in parameter W as the byte count.

The Write command uses the current track, head, and sector coordinates contained in the sector target address register (STAR).

The legal arguments for this directive are: CC, DC, SLI, NSIO.

..,WDX,..
..,WDX='arg_1,arg_2,...,arg_n',..

Inserts a Write Sector and ECC I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. This directive writes one sector of data plus four bytes which are written to the media as the ECC data. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

This directive is valid for the cache disc accelerator only.

..,WLS,..
..,WLS='arg_1,arg_2,...,arg_n',..

Inserts a Write Sector Label (WLS) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,WLT,..
..,WLT='arg_1,arg_2,...,arg_n',..

Inserts a Write Track Label I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO.

..,WR,..
..,WR,..

Inserts a Write I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic writes n number of records of parameter W bytes, or if n is not specified, the diagnostic writes the number of records specified by parameter R. If parameter W is less than four bytes, the diagnostic defaults to four bytes.

5.3.15 "Z" Directives

..,ZAP,..

This directive normally is not used if the media has been formatted. The primary purpose of the ..,ZAP,.. directive is to allow for initial formatting of the entire media. After formatting the media, all track and sector labels are tested for their validity and any errors which are detected are written to the LO file as data errors. No bad track deallocation is performed on defective track or sector labels.

Note: All tests in the diagnostic may be executed after the ..,ZAP,.. directive has been executed. The following message is written to the LO file after the program is started:

DIAGNOSTIC WILL USE MOD # DEFAULT PARAMETERS.

The following messages are associated with this directive:

YOU ARE ABOUT TO FORMAT THE DISC AND DESTROY MEDIA VERIFICATION AND POSSIBLY CUSTOMER DATA DEPRESS T TO TERMINATE. DEPRESS C TO CONTINUE

This directive is terminated by entering T (terminate). If C (continue) is entered, one of the following messages is written:

- System Disc Flag 'ON'

THE SYSTEM DISC FLAG IS ON. IF YOU WANT TO FORMAT THE DISC AND DESTROY MEDIA VERIFICATION TURN THE SYSTEM DISC FLAG OFF AND USE THE ZAP DIRECTIVE AGAIN

WHAT NEXT?

- System Disc Flag 'OFF'

THE SYSTEM DISC FLAG IS OFF!!!! TO EXIT THE DIRECTIVE DEPRESS T, TO FORMAT THE MEDIA DEPRESS C

Terminate the directive by entering T or, if the disc is being formatted, enter C to continue.

Note: Previously recorded media verification is destroyed, therefore, this directive should be used with caution.

After formatting the media and verifying labels, the program writes the following message to the listed output (LO) file:

SYSTEM DISC FLAG IS (ON)
DIAGNOSTIC WILL USE MOD # DEFAULT PARAMETERS.

..,ZIP,..

This directive reformats the diagnostic cylinders and reads all track and sector labels on the two cylinders. This directive is used prior to the execution of the diagnostic program with the system Disc Flag ON to insure that the diagnostic tracks are formatted properly.

5.4 Directives Issuing Start I/O Instructions

Directives which do not have a string argument of CC (Command Chain), DC (Data Chain), or NSIO (no Start I/O), cause the diagnostic to execute a Start I/O instruction once the directive has been processed and inserted into the IOCL queue. Directives which have string arguments of CC (Command Chain), or DC (Data Chain) insert the I/O command doubleword (IOCD) into the I/O command list (IOCL) queue and process the next directive.

5.5 Directive Arguments

Arguments may be appended to specified directives. The argument syntax is as follows:

..,directive='arg_1, arg_2,...,arg_n',..

There are six categories of arguments:

- IOCD Flag Byte Arguments
- IOCD Byte Count Arguments
- Test Sector Target Address Register (STAR) Arguments
- Load Mode Register Arguments
- Track and Sector Flag Byte Arguments
- Access Control Flag Byte Arguments

Descriptions of the six argument categories follow:

5.5.1 IOCD Flag Byte Arguments

The I/O command doubleword (IOCD) Flag Byte arguments specify modifications made to an individual IOCD within the I/O command list (IOCL) queue. IOCD Flag Byte arguments may be specified in any order. Individual directive derivatives should be consulted to define the valid IOCD Flag Byte arguments.

The IOCD Flag Byte arguments are defined as follows:

DC or D Data Chain

Sets IOCD word 2, bit 0, indicating that if the byte count equals zero, the diagnostic proceeds with the next IOCD in the IOCL queue. The next IOCD must be the same type of command as the data-chained command.

CC or C Command Chain

Sets IOCD word 2, bit 1, indicating that if the byte count equals zero, or if Channel End and Device End is detected, the diagnostic proceeds with the next IOCD in the IOCL queue. The next IOCD may be any valid IOCD.

SLI Suppress Incorrect Length Indication

Sets IOCD word 2, bit 2, indicating that Incorrect Length Status presentation should not be reported if detected.

SKIP Skip

Sets IOCD word 2, bit 3, indicating the number of bytes specified are to be skipped (not written to memory). This flag is valid only for read IOCDs.

5.5.2 IOCD Byte Count Arguments

BC(n) Fixed Byte Count (n)

Conditions the I/O command doubleword (IOCD) byte count (IOCD word 2, bits 16 through 31) to the value specified by (n). The minimum byte count for write/read operations is four bytes.

BCPW Variable Byte Count (PW)

Conditions the I/O command doubleword (IOCD) byte count (IOCD word 2, bits 16 through 31) to the value in parameter W. The minimum byte count for write/read operations is four bytes. The byte count for sector label write/read operations should be 480 bytes for a 16-sector track format and 600 bytes for a 20-sector format.

5.5.3 Test STAR Specification Arguments

TTRK(n) Test Sector Target Address Register (STAR) Track

Conditions the data address word of the Test STAR command to the track coordinate specified by (n).

THD(n) Test Sector Target Address Register (STAR) Head

Conditions the data address word of the Test STAR command to the head coordinate specified by (n).

TSEC(n) Test Sector Target Address Register (STAR) Sector

Conditions the data address word of the Test STAR command to the sector coordinate specified by (n).

NSIO No Start I/O Instruction

Inserts the current directive into the IOCL queue. This enables an IOCL queue to be built without performing a Start I/O instruction after processing the last directive.

5.5.4 Load Mode Register Arguments

The Load Mode Register (LMR) specification arguments condition the data address byte of a Load Mode Register command with the values specified in the string. If no value is given in the parentheses, the default value is used by the argument.

Read Offset Modes

The normal setting for mode register bits 0 through 3 is zero; they must be zero if a disc write operation is specified. Failure to observe this restriction causes a program check I/O program termination and a mode check (MOCK) sense buffer indication.

The offset bits provide for a normal setting (all zeros) and eight offset positions. If attempting to recover data from a removable disc pack drive, it is sometimes necessary to use the offset functions to obtain an error-free disc read operation. The procedure often used is to attempt to read three or more times at the normal position, then three or more times at both plus and minus carriage offset positions, then three or more times at both plus and minus read time offset positions, and finally three or more times at each of the four double offset positions. This process uses the four offset bits to select points in an eight-point rectangle surrounding the normal read position. Since some errors are likely to be intermittent and not repeatable, three or more tries should be made at each position.

CAR $\left\{\begin{matrix}(+)\\(0)\\(-)\end{matrix}\right\}$

Conditions the mode register byte to either set or reset the carriage offset bit. The default condition is carriage offset bit 0 reset. The subfield values within parentheses follow:

(+) Sets the carriage offset, bit 0, and resets the positive-negative offset, bit 1.

(0) Resets the carriage offset, bit 0, and resets the positive-negative offset, bit 1.

(-) Sets the carriage offset, bit 0, and sets the positive-negative offset, bit 1.

RDT $\left\{\begin{matrix}(+)\\(0)\\(-)\end{matrix}\right\}$

Conditions the Mode Register byte to either set or reset the read timing bits associated with this directive. If no values are specified in parentheses, the read timing offset bit 2 is reset. The subfield values within parentheses follow:

(+) Sets the read timing offset bit 2, and resets the positive-negative read timing strobe, bit 3.

(0) Resets the read timing offset, bit 2, and resets the positive-negative read timing strobe, bit 3.

(-) Sets the read timing offset, bit 2, and sets the positive-negative read timing strobe, bit 3.

DIAG $\left\{\begin{matrix}(ON)\\(OFF)\end{matrix}\right\}$

Conditions the Mode Register byte to either set the diagnostic mode bit 4 by using the value ON, or resets the diagnostic mode bit 4 by using the value OFF. The default condition is OFF if no subfield value is used.

MHD $\left\{\begin{matrix}(ON)\\(OFF)\end{matrix}\right\}$

Conditions the Mode Register byte to either use the moving-head disc ON or the fixed-head disc OFF for I/O operations. Bit 6 is set if the fixed-head disc is to be accessed. The default condition is moving-head disc if no subfield is used.

RES $\left\{\begin{matrix}(ON)\\(OFF)\end{matrix}\right\}$

Conditions the mode register byte to either use the reserved track mode ON by setting bit 5, or not using the reserved track mode OFF by resetting bit 5.

5.5.5 Track and Sector Flag Byte Arguments

The track and sector flag byte specification arguments specify the modification to be made to a particular track and sector label. A prototype of a sector or track label is built in the output buffer with the flag bytes set to the specified arguments. The number of prototype sector labels which are built in the output buffer depends on the number of specified bytes. However, the following byte counts are recommended:

- 600 bytes for 20-sector/track format
- 480 bytes for 16-sector format

Flag byte consistency should be maintained. A track cannot be partially designated as an alternate track and partially as a good track.

GOOD Good Track

Conditions flag byte 1, bit 0, in the output buffer and resets any other bits in the flag byte. Also, any data which is in the alternate/defective track and head field is set to zero.

ALTC(n) Alternate Track

Conditions flag byte 1 by setting the alternate track bit 1 in the output buffer and sets the defective track parameter into the appropriate slot for either track or sector labels. The parameter within the parentheses points to the associated defective track for which this track is being conditioned as an alternate track.

ALTH(n) Alternate Head

Conditions flag byte 1, bit 1, in the output buffer and sets the defective head parameter into the appropriate field for either track or sector labels. The parameter within the parentheses points to the associated defective head for which this head coordinate is being conditioned as an alternate track.

The track and head coordinates of a defective track point to the alternate track coordinates. The track and head coordinates of the alternate track must point to the defective track coordinates.

LFSC{(ON) / (OFF)} Spare Track

Conditions flag byte 1 in the output buffer by setting the spare track, bit 2.

LFRC Reserved Track

Conditions flag byte 1 in the output buffer by setting the reserved track, bit 3.

DEFC(n) Defective Track Cylinder

Conditions flag byte 1 by setting the defective track, bit 4, in the output buffer and sets the alternate track parameter into the appropriate field for either track or sector labels. The parameter in the parentheses (n) points to the alternate track.

DEFH(n) Defective Head

Conditions flag byte 1 by setting the defective track, bit 4, in the output buffer and sets the alternate head parameter into the appropriate field for either track or sector labels. The parameter in the parentheses (n) points to the alternate head.

LAST{(ON) / (OFF)} Last Track

Conditions flag byte 1 in the output buffer by setting the last track bit, 5.

FLOK{(ON) / (OFF)} Write Lock

Conditions flag byte 2 in the output buffer by setting the write lock bit, 0, for either track or sector labels. Refer to Access Control Flag Byte Arguments.

FPRO{(ON) / (OFF)} Write Protect

Conditions flag byte 2 in the output buffer by setting the write protect bit, 1, for sector labels only. Refer to Access Control Flag Byte Arguments.

5.5.6 Access Control Flag Byte Arguments

The Access Control Flag Byte (track label flag byte 2) arguments specify modifications made to a particular track and sector label. The byte contains the Write Protect bit and the Write Lock bit; the remaining six bits must be zero. The arguments are valid for the following directives only:

..,LLS='arg_1,arg_2,...,arg_n',..	Load Sector Label
..,LLT='arg_1,arg_2,...,arg_n',..	Load Track Label
..,RLS='arg_1,arg_2,...,arg_n',..	Read Sector Label
..,RLT='arg_1,arg_2,...,arg_n',..	Read Track Label
..,WLS='arg_1,arg_2,...,arg_n',..	Write Sector Label
..,WLT='arg_1,arg_2,...,arg_n',..	Write Track Label

5.6 Parameter Descriptions

The Gould RPU Disc Processor Diagnostic uses the following set of 26 unique parameters (identified A through Z) which are manipulated by the P directive:

Parameter	Use	Description
A	Reserved	Used by the Diagnostic Executive Program (DEXP).
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word 1 data pattern if the output buffer word one has been set using a ..,PD( )k,.. directive.
E	Reserved	Contains the number of sectors on the disc drive (sector modulus).

Parameter	Use	Description
F	Reserved	Contains the number of heads on the disc drive (head modulus). Refer to Disc Drive Test Parameters for the values and system disc flag settings associated with this parameter.
G	Reserved	Contains the number of cylinders on the disc drive (cylinder modulus). This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Test Parameters for the values and System Disc Flag settings associated with this parameter.
H	Reserved	Contains the current head address coordinate.
I	Reserved	Contains the current output buffer word 1 data pattern if the output buffer word 1 had been set using a ..,PI( )k,.. directive.
J	Reserved	Contains the highest valid sector address (parameter E minus 1).
K	Reserved	Contains the highest valid head address (parameter F minus 1).
L	Reserved	Contains the starting cylinder address to be used by the diagnostic. The starting cylinder address defaults to the diagnostic starting cylinder during program initialization. Refer to Disc Drive Test Parameters for the values and system disc flag settings associated with this parameter.
M	Available	Available for general use.
N	Reserved	Contains the number of bytes to use if operating in the diagnostic mode.
O	Reserved	The highest valid cylinder that normally is tested if the System Disc Flag is OFF. This value normally is set to parameter G minus 4. This value always is inserted into parameter U. The use of data parameter O as the ending cylinder prevents the diagnostic from writing over the media certification data which is on cylinder PG minus 4 or less, as determined by Media Verification.
P	Reserved	Contains the current track, head, and sector address coordinates: B0 through 15 — track address coordinate B16 through 23 — head address coordinate B24 through 31 — sector address coordinate

Parameter	Use	Description
Q	Reserved	Contains the cylinder address of the designated diagnostic track. This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Default and Test Parameters for the values and system disc flag settings associated with this parameter.
R	Reserved	Contains the record count for ..,WR,.. and ..,RR,.. directives.
S	Reserved	Contains the current sector address coordinate.
T	Reserved	Contains the current track address coordinate.
U	Reserved	Contains the highest valid cylinder address to be used by the diagnostic. This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Default and Test Parameters for the values and system disc flag settings associated with this parameter.
V	Reserved	Contains the number of bytes in one sector.
W	Reserved	Contains the implicit transfer count for write and read data directives.
X	Reserved	Contains the default hold duration for ..,H,.. and ..,HS,.. directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for ..,JZ,.. directives.

5.7 Disc Drive Default and Test Parameters

The tables enclosed in this section show the value of the parameters used by the program if tests are performed under the specified conditions. The controlling factors for the diagnostic are the model number, system disc flag setting and dual or single port drive. A brief description of the test sequence(s) is given and some tests are skipped automatically if certain conditions are detected by the diagnostic. An example of the directive string needed for the particular mode of operation is also given.

Parameter	Description
F	The maximum number of heads on the drive.
G	The maximum number of cylinders on the drive.
K	The highest valid head address.
L	The starting cylinder used by the diagnostic.

Parameter	Description
O	The highest valid cylinder that is tested by the diagnostic. The default value is always equal to parameter G minus 5. This value always is inserted into parameter U. The use of parameter O as the ending cylinder prevents the diagnostic from writing over the media certification data which is on cylinder parameter G minus 4.
Q	The starting and/or ending diagnostic cylinder coordinates. Cylinders PG minus 3 and PG minus 2 are the default parameters dedicated to the diagnostic cylinders.
U	The highest valid testable track used by the diagnostic. This value is dependent on the mode of operation, System Disc Flag setting, and single or dual port.

The default parameters are determined during the execution of the Gould RPU Media Verification Program or upon completion of the ..,ZAP,.. directive in the diagnostic program.

However, the Gould RPU Media Verification program may move the diagnostic cylinder or the ending test cylinder if the initial parameter settings for the deallocated track information or the diagnostic track are detected to be faulty because of address mark errors, disc format errors or No System Interrupt Receive from an I/O command.

The diagnostic cylinder parameters and end of media for the diagnostic are written to the LO file using the ..,DBT,.. directive.

The default parameter values associated with the AU, MOD, DUAL1, DUAL2, DUAL0, and SING directives are described in the following sections:

5.7.1 System Disc Flag is Set ON, Drive is in Single Port Mode

Drive Model #	Parameters F	G	K	L	O	Q	U
8104 (Fixed Head)	4	64	3	61	59	61-62	62
8117 (Cartridge)	1	823	0	820	818	820-821	821
Removable 8117 (Cartridge) Captive	1	823	0	820	818	820-821	821
8122 (Winchester)	5	823	4	820	818	820-821	821
8127 (Winchester)	10	823	9	820	818	820-821	821
8136 (Cartridge 9in)	5	823	4	820	818	820-821	821
8146 (Winchester 9in)	10	823	9	820	818	820-821	821
8155 (Winchester)	40	843	39	840	838	840-841	841
8420 (Winchester 8in)	5	823	4	820	818	820-821	821
8424 (Winchester 8in)	8	1024	7	1021	1019	1021-1022	1022
8858 (Winchester 9in)	24	711	23	700	706	708-709	709
9342 (Removable)	5	823	4	820	818	820-821	821
9344 (Removable)	19	823	18	820	818	820-821	821

Directive Example

DXP>D1 AU0,MOD9344, SDF='SYS(ON)'

Note: The model number defaults the drive to system flag setting of ON, single ported and controller Model 8055.

Test Execution

Tests 1 through 41 are executed on the diagnostic cylinders for whichever model number is selected with the following exceptions:

- Test 4 reads all labels on the diagnostic cylinder and reports any errors detected as data errors.
- Test 26 performs only random seek and read sector labels operations. No data is written to the media.
- Tests 38 and 39 are skipped if option 18 (Enables Tests 38 and 39) is reset.
- Test 40 is skipped if the Disc Processor Model 8055 is entered prior to test execution by using the ..,CON8055,.. directive.

5.7.2 System Disc Flag is Set OFF, Drive is in Single Port Mode

Drive Model #	Parameters F	G	K	L	O	Q	U
8104 (Fixed Head)	4	64	3	0	59	61-62	59
8117 (Cartridge) Removable	1	823	0	0	818	820-821	818
8117 (Cartridge) Captive	1	823	0	0	818	820-821	818
8122 (Winchester)	5	823	4	0	818	820-821	818
8127 (Winchester)	10	823	9	0	838	820-821	818
8136 (Cartridge 9in)	5	823	4	0	818	820-821	818
8146 (Winchester 9in)	10	823	9	0	838	820-821	818
8155 Winchester)	40	843	39	0	838	840-841	838
8420 (Winchester 8in)	5	823	4	0	818	820-821	818
8424 (Winchester 8in	8	1024	7	0	1019	1021-1022	1019
8858 (Winchester 9in)	24	711	23	0	706	708-709	706
9342 (Removable)	5	823	4	0	818	820-821	818
9344 (Removable)	19	823	18	0	818	820-821	818

Directive Example

DXP>D1 AU2, MOD9344, SDF='SYS(OFF)', SING

Test Execution

Tests 1 through 40 are executed using the entire media, where appropriate, beginning with track zero and ending with the last testable track which is listed under parameter O, with the following execeptions:

- Test 26 performs random seek and write single sectors followed by reseeking the sectors which are accessed and reading the data which is written.
- Tests 38 and 39 are skipped if option 18 (Enables Tests 38 and 39) is reset.

5.7.3 System Disc Flag is Set ON, Drive is Set to Dual Port by DUAL1 Directive

Drive Model #	Parameters F	G	K	L	O	Q	U
8122 (Winchester)	5	823	4	820	820	820	820
8127 (Winchester)	10	823	9	820	820	820	820
8136 (Cartridge 9in)	5	823	4	820	820	820	820
8146 (Winchester 9in)	10	823	9	820	820	820	820
8155 (Winchester)	40	843	39	840	840	840	840
8858 (Winchester 9in)	24	711	23	708	708	708	708
9342 (Removable)	5	823	4	820	820	820	820
9344 (Removable)	19	823	18	820	820	820	820

Note: The starting track parameter L, the diagnostic parameter Q and the ending cylinder parameter U are equal to the first cylinder of the two diagnostic cylinders. The first diagnostic cylinder is tested by the diagnostic.

Directive Example

```
DXP>D1 AU4, MOD8117, SF='SYS(ON)',DUAL1
```

Test Execution

Tests 1 through 40 are executed on the first allocated diagnostic cylinder, with the following exceptions:

- Tests 3, 6 through 9, 12, 25, and 37 are skipped if option 20 (conditions the diagnostic to run those tests) is reset.
- Test 4 reads all labels on the media and reports all flag byte label settings detected by the diagnostic.
- Test 9 automatically is skipped because it performs defective and alternate track tests on the other diagnostic cylinder. This test should be executed by itself by setting option 20 (conditions the diagnostic to run Test 9). Each CPU should execute the test individually.
- Test 26 performs random seek and read sector label operations. No write data transfer operations are performed by this test.
- Tests 38 and 39 will be skipped if option 18 (Enables Tests 38 and 39) is reset.

5.7.4 System Disc Flag is Set ON, Drive is Set to Dual Port by DUAL2 Directive

Drive Model #	Parameters F	G	K	L	O	Q	U
8122 (Winchester)	5	823	4	821	821	821	821
8127 (Winchester)	10	823	9	821	821	821	821
8136 (Cartridge 9in))	5	823	4	821	821	821	821
8146 (Winchester 9in)	10	823	9	821	821	821	821
8155 (Winchester)	40	843	39	811	841	841	841
8858 (Winchester 9in)	24	711	23	709	709	709	709
9342 (Removable)	5	823	4	821	821	821	821
9344 (Removable)	19	823	18	821	821	821	821

Note: The starting cylinder parameter L, the diagnostic parameter Q, and the ending cylinder parameter U are equal to the second cylinder of the two diagnostic cylinders. The second diagnostic cylinder is tested by the diagnostic.

Directive Example

```
DXP>D1 AU0, MOD8055, SDF='SYS(ON)',DUAL2
```

Test Execution

Tests 1 through 40 are executed on the second allocated diagnostic cylinder with the following exceptions:

- Tests 3, 6 through 9, 12, 25 and 37 are skipped if option 20 (conditions the diagnsotic to execute those tests) is reset.
- Test 4 reads all labels on the disc and reports all flag byte label settings detected by the diagnostic.
- Test 9 automatically is skipped as it performs defective and alternate track tests on both diagnostic cylinders. This test should be executed by itself by setting option 21 (conditions the diagnostic to run Test 9). Each CPU should execute the test individually.
- Test 26 performs random seek and read sector label operations. No write data transfer operations are performed by this test.
- Tests 38 and 39 will be skipped if option 18 (Enables Tests 38 and 39) is reset.

5.7.5 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL1 Directive

Drive Model #	Parameters F	G	K	L	O	Q	U
8122 (Winchester)	5	823	4	0	407	408	409
8127 (Winchester)	10	823	9	0	407	408	409
8136 (Cartridge 9in)	5	823	4	0	407	408	409
8146 (Winchester 9in)	10	823	9	0	407	408	409
8155 (Winchester)	40	843	39	0	415	415	415
8858 (Winchester 9in)	24	711	23	0	349	349	349
9342 (Removable)	5	823	4	0	407	408	409
9344 (Removable)	19	823	18	0	405	405	405

Directive Example

```
DXP>D1 AU0, MOD9342, SDF='SYS(OFF)', DUAL1
```

Note: The first half of the media is tested by the diagnostic program.

Test Execution

- Tests 3, 6 through 9, 12, 25 and 37 are skipped if option 20 (conditions the diagnostic to execute those tests) is reset.
- Tests 1 through 40 are performed using the entire allocated media, where appropriate, beginning with track zero and ending with cylinder parameter O.
- Test 4 formats the disc and reads labels beginning with the parameter specified in PL and ending with the cylinder parameter specified in PO.
- Tests 38 and 39 are skipped if option 18 (Enables Tests 38 and 39) is reset.

5.7.6 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL2 Directive

Drive Model #	Parameters F	G	K	L	O	Q	U
8122 (Winchester)	5	823	4	409	818	818	818
8127 (Winchester)	10	823	9	409	818	818	818
8136 (Cartridge 9in)	5	823	4	409	818	818	818
8146 (Winchester 9in)	10	823	9	409	818	818	818
8155 (Winchester)	40	843	39	419	838	840	838
8858 (Winchester 9in)	24	711	23	353	706	708	706
9342 (Removable)	5	823	4	409	818	818	818
9344 (Removable)	19	823	18	409	818	820	818

Directive Example

```
DXP>D1 AU0, MOD9342, SDF='SYS(OFF)',DUAL2
```

Note: The second half of the media is tested by the diagnostic program.

Test Execution

- Tests 3, 6 through 9, 12, 25 and 37 are skipped if option 20 (conditions the diagnostic to execute those tests) is reset.
- Tests 1 through 40 are performed using the allocated cylinders, where appropriate, beginning with cylinder PL and ending with cylinder parameter O.
- Test 4 reads labels beginning with the cylinder zero and ending with the cylinder parameter specified in O.
- Tests 38 and 39 are skipped if option 18 (Enables Tests 38 and 39) is reset.

5.7.7 System Disc Flag is Set OFF, Drive is Set to Dual Port by DUAL0 Directive.

Drive Model #	Parameters F	G	K	L	O	Q	U
8122 (Winchester)	5	823	4	0	818	820	818
8127 (Winchester)	10	823	9	0	818	820	818
8136 (Cartridge 9in)	5	823	4	0	818	820	818
8146 (Winchester 9in)	10	823	9	0	818	820	818
8155 (Winchester)	40	843	39	0	838	840	838
8858 (Winchester 9in)	24	711	23	0	06	08	706
9342 (Removable)	5	823	4	0	818	820	818
9344 (Removable)	19	823	18	0	818	820	818

Directive Example

```
DXP>D1 AU0, MOD9342,SDF='SYS(OFF)',DUAL0
```

Test Execution

- Tests 1 through 40 are performed using the entire media, where appropriate, beginning with track zero and ending with cylinder parameter O.
- Tests 4 reads labels beginning with cylinder zero and ending with the cylinder specified in parameter O.
- Tests 38 and 39 are skipped if option 18 (Enables Tests 38 and 39) is reset.

CHAPTER 6

I/O COMMAND TRACE

6.1 Tracing I/O Activity

The Gould RPU Disc Processor Diagnostic contains optional capabilities that provide visual representation of I/O activity used to augment error message reports and to develop a better understanding of Disc Controller protocol.

If option 17 (Writes I/O Command Trace) is set, the I/O Command Trace records all Start I/O instructions performed on the units under test to the listed output (LO) file.

6.2 I/O Command Trace

The I/O Command Trace consists of:

1. The task separation bannner

```
<<<<'taskname '----hh:mm:ss>>>>
```

2. The test header

```
IOCLQP:          aaaaaaaa
IOCD(#)       (1) oobbbbbb      (2) ffOOcccc
                    .
                    .
                    .
                  etc.
```

aaaaaaaa	specifies the real main memory address of the I/O command list (IOCL) address.
#	specifies the relative I/O command list IOCD decimal slot number in the IOCL Queue.
oo	specifies the contents of the order byte (IOCD word 1, byte 0).
bbbbbb	specifies the real main memory buffer address.
ff	specifies the contents of the flag byte (IOCD word 2, byte 0).
cc	specifies the hexadecimal byte count.

Gould RPU High-Speed Tape Processor

(RP.HTP.r)

Peripheral Diagnostic Description

Diagnostic Number: 001032-942r

Documentation Release Level: 1.0

Documentation Release Date: September 1984

Publication Order Number: 326-005300-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1984
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould RPU High-Speed Tape Processor (RP.HTP.r) Peripheral Diagnostic Description, Publication Order Number **326-005300-000,** was printed September 1984.

This manual contains the following pages:

Title page
Copyright page
History page
iii through v/vi
1-1/1-2
2-1 through 2-3/2-4
3-1 through 3-37/3-38
4-1 through 4-12
5-1 through 5-24

CONTENTS

Chapter Page

1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1

2 ASSIGNMENTS AND EXECUTION

2.1 Device Assignments 2-1
2.2 Model Number Assignments 2-1
2.3 Diagnostic Option Set 2-1
2.4 Diagnostic Execution 2-3
2.5 Milestone Messages 2-3

3 TEST DESCRIPTIONS

3.1 Summary of Tests 3-1
3.2 Test 1--Interrupt Instruction Verification 3-1
3.3 Test 2--Service Interrupt Verification and Static Halt I/O, Stop I/O, Test I/O 3-1
3.4 Test 3--Quiescent Unit Status Verification 3-1
3.5 Test 4--Controller Program Violation 3-2
3.6 Test 5--Basic Tape Motion 3-2
3.7 Test 6--Creep/Erase 3-2
3.7.1 Subtest A 3-2
3.7.2 Subtest B 3-3
3.7.3 Subtest C 3-3
3.7.4 Subtest D 3-3
3.8 Test 7--Residual Transfer Count and Incorrect Length 3-3
3.8.1 Subtest A 3-3
3.8.2 Subtest B 3-3
3.8.3 Subtest C 3-4
3.8.4 Subtest D 3-4
3.8.5 Subtest E 3-4
3.8.6 Subtest F 3-4
3.9 Test 8--Invalid Memory Access 3-4
3.10 Test 9--Program Violation 3-5
3.11 Test 10--IOCL Termination and File Mark Detection 3-5
3.12 Test 11--Interface Program Violation 3-5
3.13 Test 12--Rewind (Variable Length) 3-5
3.14 Test 13--Write/Backspace/Write File Marks 3-5

Chapter		Page
3.15	Test 14--Write/Backspace/Advance File Marks Command Chained	3-6
3.16	Test 15--Write/Backspace/Advance File Marks (Variable Gap Length)	3-6
3.17	Test 16--File Mark Rejection	3-7
3.18	Test 17--Write/Backspace/Write Record	3-7
3.19	Test 18--Write/Backspace/Advance Record	3-7
3.20	Test 19--Write/Backspace/Read Record	3-8
3.21	Test 20--Write/Rewind/Read Record	3-8
3.22	Test 21--Write/Rewind/Read Record (Variable Gap Length)	3-8
3.23	Test 22--Write/Backspace/Read Record (Variable Gap Length)	3-9
3.24	Test 23--Write/Rewind/Read Record (Fixed Length/Fixed Data)	3-10
3.25	Test 24--Write/Backspace/Read Record (Variable Length/Fixed Data)	3-11
3.26	Test 25--Write/Backspace/Read (Blocked Files)	3-11
3.27	Test 26--Write/Record (Random Length)	3-12
3.28	Test 27--Read Record (Random Length)	3-13
3.29	Test 28--Write/Read Backward Record (Variable Length/Fixed Data)	3-13
3.30	Test 29--Write/Backspace/Read Compare (Variable Length/Fixed Data)	3-14
3.31	Test 30--Two Data Chain Read	3-14
3.32	Test 31--Four Data Chain Read	3-15
3.33	Test 32--Skip Read Data Chained	3-16
3.34	Test 33--Two Data Chain Write	3-17
3.35	Test 34--Four Data Chain Write (Variable Length)	3-18
3.36	Test 35--Read Compare Data Chained	3-19
3.37	Test 36--Four Data Chain Read Compare	3-19
3.38	Test 37--Skip Read Compare Data Chained	3-21
3.39	Test 38--Four Data Chain Skip in Second and Fourth Read Compare	3-21
3.40	Test 39--Read Backward Data Chained	3-23
3.41	Test 40--Four Data Chained Read Backward	4-23
3.42	Test 41--Read Backward Skip Data Chained	3-24
3.43	Test 42--Variable Buffer Starting Address	3-25
3.44	Test 43--Post Program Control Interrupt	3-26
3.44.1	Subtest A	3-26
3.44.2	Subtest B	3-26
3.44.3	Subtest C	3-27
3.44.4	Subtest D	3-27
3.44.5	Subtest E	3-27
3.45	Test 44--Halt I/O, Stop I/O, Test I/O Dynamic State	3-28
3.45.1	Subtest A	3-28
3.45.2	Subtest B	3-28
3.45.3	Subtest C	3-28
3.45.4	Subtest D	3-28
3.45.5	Subtest E	3-28
3.45.6	Subtest F	3-28
3.46	Test 45--Scatter/Gather Write	3-29
3.47	Test 46--Scatter/Gather Read	3-30

Chapter Page

3.48 Test 47--Scatter/Gather Read Compare 3-32
3.49 Test 48--Sync Chase Initial 3-33
3.50 Test 49--Data Chain Sync Chase 3-34
3.51 Test 50--Variable Sync Chase 3-35

4 ERROR REPORTING

4.1 Error Message Assignment 4-1
4.2 Error Messages and Message Sequence 4-1
4.3 Status Error Report Exceptions 4-2
4.4 Condition Code Errors 4-2
4.5 No Interrupt or Device End Received Error Message 4-3
4.6 Status Address Word Error 4-4
4.7 Program Start Errors 4-4
4.8 Data Error Reporting 4-5
4.9 Additional Error Messages 4-8
4.10 Formatter Reject Code Messages 4-10

5 DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Compiler 5-1
5.2 DEXP Standard Directive 5-1
5.2.1 "A" Directives 5-1
5.2.2 "C" Directives 5-2
5.2.3 "D" Directives 5-2
5.2.4 "H" Directives 5-3
5.2.5 "J" Directives 5-3
5.2.6 "L" Directives 5-4
5.2.7 "P" Directives 5-6
5.2.8 "S" Directive 5-7
5.2.9 "T" Directive 5-7
5.3 Gould RPU High-Speed Tape Processor Unique Directives 5-7
5.3.1 "A" Directives 5-8
5.3.2 "B" Directives 5-8
5.3.3 "C" Directives 5-9
5.3.4 "D" Directives 5-10
5.3.5 "E" Directive 5-11
5.3.6 "I" Directive 5-12
5.3.7 "L" Directives 5-12
5.3.8 "N" Directive 5-13
5.3.9 "R" Directives 5-13
5.3.10 "S" Directives 5-15
5.3.11 "T" Directives 5-16
5.3.12 "W" Directives 5-16
5.4 Directives Issuing Start I/O Instructions 5-17
5.5 Directive Arguments 5-17
5.5.1 IOCD Flag Byte Arguments 5-17
5.5.2 IOCD Byte Count Argument 5-18
5.5.3 Mode Register Arguments 5-19
5.5.4 DUMP Directive Arguments 5-21
5.6 Directive Error Reporting 5-23
5.7 Parameter Descriptions 5-24

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould RPU High-Speed Tape Processor Diagnostic (RP.HTP.r) is a Level Two program which loads and executes under control of the Diagnostic Executive Program (DEXP). The DEXP provides a multiprogrammed environment allowing simultaneous execution of any number of Level Two diagnostics (limited to 512K bytes of memory) while providing direct operator control for fault isolation and troubleshooting.

The diagnostic contains a sequence of 50 tests, which verify the correct operation of a High-Speed Tape Processor function. The tests execute sequentially (in an automatic mode) or individually (by operator request).

The diagnostic requires a minimum of operator intervention. Tests requiring operator intervention may be skipped.

During execution, the diagnostic writes milestone messages and error message sequences to the listed output (LO) file, which is assigned to the operator's console or a line printer.

The diagnostic's peripheral language compiler and run-time support package provide:

- The ability to create new directive test sequences.
- The ability to reduce diagnostic test sequences into their failing components for troubleshooting.

The diagnostic reads operator-composed directive sequences from the command control (CC) file (if in batch mode) or from the operator communications (OC) file (if in interactive mode). The CC file is assigned to disc or tape and the OC file is assigned to the operator's console or a terminal configured on an Eight-Line Asynchronous Controller.

1.2 Resource Requirements

The Gould RPU High-Speed Tape Processor Diagnostic requires:

- A Gould CPU with a minimum of 64K words of memory.
- A Tape Processor (Model 8050).

CHAPTER 2

ASSIGNMENTS AND EXECUTION

2.1 Device Assignments

The Gould RPU High-Speed Tape Processor Diagnostic defaults to channel address 10_{16}, device subaddress 00_{16}.

To modify the default channel address and/or device subaddress(es), enter:

DXP>ASSIGN taskname,MTaa,dsa_1,dsa_2,...,dsa_n

taskname — specifies the one to eight character name indicated in the load map.

MTaa — indicates a four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.

dsa — specifies the device subaddress(es) to be assigned (maximum of eight device subaddresses designated 00 through 07_{16}).

2.2 Model Number Assignments

The Gould High-Speed Tape Processor does not require model number assignments because all model numbers are functionally alike. Initial program load and any system reset conditions default the diagnostic to the Group Coded Recording (GCR) mode of operation.

2.3 Diagnostic Option Set

The Gould RPU High-Speed Tape Processor Diagnostic is a self-initializing program, therefore, it issues no initialization prompts after loading. To load the diagnostic, follow the procedures in Chapter 7, Level Two Diagnostics, of the Gould Virtual Memory Diagnostic Facility User's Manual. The program name for loading the diagnostic is RP.HTP.r. After the diagnostic loads, it enters a wait state until receiving directions through the Gould Diagnostic Executive Program (DEXP).

Using the DEXP, select diagnostic options to modify default execution conditions.

The Gould RPU High-Speed Tape Processor Diagnostic contains the following options:

Options	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If the option is set during an error message report, the message is completed. Option 1 is not inhibited by this option.

Options	Default	Effect When Set
1	Reset	Rings the bell on the operator's console once per error.
2	Reset	Suspends diagnostic execution if an error is detected. Prior to suspension, the associated error message is written, unless option 0 is set. During suspension, the following message is written: 'taskname' ERROR... ;[C,H,T]? The query [C,H,T]? requires a response. Responding with C, continues execution. Responding with T, terminates execution. Responding with H, places the diagnostic on hold, allowing time for setting options, performing memory dumps, or manipulating other diagnostics running concurrently. Nullify the hold state with a Continue command.
3	Reset	Repeats the sequence detecting an error until obtaining a correct result. During the repeat function, no operand values change. If options 0, 1, and 2 are set, their functions occur prior to this function.
4	Reset	Repeats the current subtest until the option is reset.
5	Reset	Repeats the current test until the option is reset.
6	Set	Repeats the entire diagnostic until the option is reset.
7	Reset	Runs only tests coded in directive language.
8	Reset	Inhibits tests requiring operator intervention.
9	Reset	Inhibits condition code and no status interrupt error messages.
10	Set	Writes the diagnostic's pass count messages to the listed output (LO) file.
11	Reset	Writes the diagnostic's milestone messages to the listed output (LO) file.
12	Reset	Inhibits error message expansion of both Status and Sense data.
13	Reset	Inhibits Status error messages.
14	Reset	Inhibits Sense error messages.
15	Reset	Inhibits Data error messages.
16	Reset	Inhibits automatic mode change.
17	Reset	Enables I/O command trace.
18	Reset	Inhibits writing error bit analysis messages.

2.4 Diagnostic Execution

After setting options, issue a Start command to begin diagnostic execution. As the diagnostic begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-942r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the diagnostic begins testing and writes milestone messages to the OC file.

2.5 Milestone Messages

The diagnostic verifies the presence of the I/O channel and if the channel is absent or not receptive to commands, the diagnostic writes the following message to the operator communications (OC) file:

```
'taskname' TERMINATED: CANNOT INITIALIZE CHANNEL
```

Reassign the device to be tested, using the correct channel address. If more than one device subaddress is assigned, the diagnostic resets the device address table pointer to start testing on the first assigned device subaddress. Each test is executed sequentially until all tests are executed.

If option 11 (writes milestone messages) is set, the following message is written to the listed output (LO) file prior to execution of each test:

```
hh:mm:ss 'taskname' TEST n (test description)
```

After testing the current device subaddress, the device subaddress table pointer is advanced to the next device subaddress to be tested. The sequence is repeated for this device subaddress and each subsequent device subaddress.

If option 10 (writes pass count messages) is set, the following message is written to the listed output (LO) file after all tests on all device subaddresses assigned is completed:

```
hh:mm:ss 'taskname' FINISHED PASS n
```

The diagnostic then resets the device subaddress table pointer and repeats the sequence until the diagnostic is commanded to terminate.

CHAPTER 3

TEST DESCRIPTIONS

3.1 Summary of Tests

The Gould RPU High-Speed Tape Processor Diagnostic contains a sequence of 50 tests which run in a continuous sequence or individually by operator request.

Test sequences 1 through 11 verify the correct functioning of various High-Speed Tape Processor functions and exceptional conditions. Tests 12 through 50 verify device mechanism functions.

Where noted, test sequences are skipped by selecting the appropriate options.

3.2 Test 1--Interrupt Instruction Verification

This test verifies that the basic interrupt control instructions function properly. The following instructions are tested: Disable Channel Interrupt (DCI), Enable Channel Interrupt (ECI), Activate Channel Interrupt (ACI), Deactivate Channel Interrupt (DACI), Block External Interrupts (BEI), Unblock External Interrupts (UEI), and the I/O instructions Start I/O (SIO) and Test I/O (TIO).

3.3 Test 2--Service Interrupt Verification and Static Halt I/O, Stop I/O, Test I/O

This test verifies that the High-Speed Tape Processor generates the following:

- a service interrupt and post correct sense data as a result of a Start I/O instruction.
- a service interrupt when a Halt I/O or Stop I/O instruction is executed while a drive is not active.

In addition, Test 2 verifies that the Test I/O instruction functions properly.

3.4 Test 3--Quiescent Unit Status Verification

This test verifies that the High-Speed Tape Processor detects the following status and sense data bits: channel end, device end, unit exception, unit check, interface check, file protect, beginning-of-tape, unit ready, unit-on-line, and intervention required. Messages are written to the operator communications (OC) file to aid in performing the test properly.

This test requires operator intervention and is skipped if option 8 (Inhibits operator intervention tests) is set.

3.5 Test 4--Controller Program Violation

This test verifies that the High-Speed Tape Processor detects status of Program Check (bit 3) and a sense of Command Reject (bit 0) if commanded to perform an undefined function (illegal order). This check is made for all undefined functions.

3.6 Test 5--Basic Tape Motion

RWW,WR1,WFMI,AF1,BF1,BR1,AR1,RR1,WR1,RC,RB,:

This test verifies that the High-Speed Tape Processor performs a Start I/O instruction to rewind and wait for BOT, write, write file mark, advance file mark, backspace file mark, backspace record, advance record, read, write, read compare, read backward, and return status consistent with the operation performed.

Each command is inserted into the I/O command list (IOCL) queue and is initiated by issuing a Start I/O instruction. After completing each command, a Sense command is executed and the sense data is checked to verify that it is consistent with what is expected.

This test also command chains the instruction in the IOCL queue, performs a Start I/O instruction and verifies that the command chaining is functioning.

3.7 Test 6--Creep/Erase

This test verifies that the High-Speed Tape Processor performs a write/backspace, write/read backward, write/backspace/read compare sequence with no reverse creep indication. The test is composed of four subtests.

3.7.1 Subtest A

1. RW,PW=V,LBP,1,WR1,ER1,BR1,J1/30,..

   This subtest rewinds the device mechanism, defaults the byte count to 4096 bytes per record, loads a binary progression, writes a record, erases one record, backspaces a record, and continues the sequence 30 times.

2. ..,WR1,ER1,WFM1,..

   The test then writes a record, erases one record, and writes a file mark.

3.7.2 Subtest B

This subtest rewinds the device mechanism and advances two records, verifying that the file mark from subtest 6A is detected, then rewinds the device.

3.7.3 Subtest C

..,1,WR1,RB,J1/30,..

This subtest rewinds the device mechanism, defaults the byte count by parameter W to 4096 bytes per record, writes a record, reads a record backward, and continues the sequence 30 times.

3.7.4 Subtest D

..,1,WR1,BR1,RC,J1/30,:

This subtest rewinds the device mechanism, defaults the byte count by parameter W to 4086 characters per record, writes a record, backspaces a record, performs a read compare, and continues the sequence 30 times.

3.8 Test 7—Residual Transfer Count and Incorrect Length

This test verifies that the High-Speed Tape Processor detects a remaining byte count by a status of Incorrect Length and that the Incorrect Length status is suppressed by setting the Suppress Incorrect Length flag bit in a read command. The test is composed of six subtests.

3.8.1 Subtest A

RW,LBP,W2048,RW,R2048,..

This subtest rewinds the device mechanism, loads a binary progression, writes one record of 2048 bytes, rewinds the device mechanism, and reads one record of 2048 bytes.

3.8.2 Subtest B

..,RW,R1024,..

This subtest rewinds the device mechanism, reads the record written in subtest 7A using a byte count of 1024, and tests for an Incorrect Length.

3.8.3 Subtest C

..,RWW,R4096,..

This subtest rewinds the device mechanism, reads the record written in subtest 7A using a byte count of 4096, and tests for an Incorrect Length and a remaining byte count of 800_{16}.

3.8.4 Subtest D

..,RWW,R='BC(4096),SLI',..

This subtest rewinds the device mechanism, reads the record written in subtest 7A using a byte count of 4096 and a flag byte indicative Suppress Incorrect Length status, and then tests for a residual byte count of 800_{16}.

3.8.5 Subtest E

This verifies that the High-Speed Tape Processor sets the Illegal Length and Unit Exception status (bits 2 and 15) along with Device End and Channel End. A residual byte count of 4096 is expected along with Sense data which shows Ready, Online, and Beginning-of-Tape (BOT).

3.8.6 Subtest F

This test verifies that the High-Speed Tape Processor posts residual byte counts starting with 32 and decrementing to two if Read, Read Backward, and Read Compare commands are executed over a two-byte record. The test starts by writing a two-byte record to the device. The diagnostic then starts with a byte count of 34, performs the Read commands and determines that the correct residual byte count is posted along with an Illegal Length status (bit 2). The program decrements the byte count by one and continues the testing until the byte count is decremented to four bytes.

3.9 Test 8--Invalid Memory Access

This test verifies that the High-Speed Tape Processor (HTP) detects an invalid memory access condition if requested to transfer data from a nonpresent memory module. The HTP responds with a status indication of Channel Program Check. The diagnostic verifies that no data is transferred by checking that the residual byte count is equal to the number of bytes that are to be written.

For systems with 16 MB of memory (no nonpresent memory module in the test range), this test is skipped automatically following the test identification milestone message.

3.10 Test 9--Program Violation

This test verifies that the High-Speed Tape Processor detects the following program violations:

- Transfer-In-Channel (TIC) as the first command in the I/O command list (IOCL).
- Transfer-In-Channel (TIC) to a Transfer-In-Channel (TIC).
- TIC with the data address pointing to a byte and halfword boundary.
- A Write File Mark is data chained.
- A Write command with a byte count of seven is issued and the drive is in the Scatter/Gather mode of operation.

The High-Speed Tape Processor responds to the above violations with a status indication of Control Program Check (status bit 3). The program also checks for sense data of Ready, Online, and Beginning-of-Tape (BOT).

3.11 Test 10—IOCL Termination and File Mark Detection

This test verifies basic motion commands and that the status word correctly points to the proper address if command chain or data chain for Read, Read Backward, and Read Compare commands are executed over a file mark. This test is composed of 15 subtests.

3.12 Test 11—Interface Program Violation

This test verifies that an unsupported transaction condition code (5) setting is detected if the High-Speed Tape Processor attempts to execute Write Channel WCS (WWCS), Grab Controller (GRIO), or Reset Controller (RSCTL) commands.

3.13 Test 12—Rewind (Variable Length)

The test sequence follows:

1. RWW,PY=64,1,WFM1,ER,RW,..

   Rewinds the device, defaults the erase iteration parameter Y to 64, performs one filemark to advance the device off-load point and parameter Y number of erases to move the tape forward, and rewinds the device.

2. ..,PY/2,J1/6,:

   Then the test divides parameter Y by two and repeats the write/erase/rewind sequence until rewinds are performed from distances varying from 18 feet to 3.5 inches off-load point.

3.14 Test 13—Write/Backspace/Write File Marks

The test sequence follows:

1. RWW,1,WFM1,BF1,WFM1,J1/24,..

   Rewinds the device mechanism, writes one file mark, backspaces one file mark, rewrites the file mark, and repeats the write file mark/backspace a file/write a file mark sequence until 25 file marks are written.

2. ..,BF25,AF25,:

   The test verifies the existence of the 25 file marks by backspacing 25 file marks and advancing 25 file marks.

3.15 Test 14--Write/Backspace/Advance File Marks Command Chained

The test sequence follows:

1. RWW,1,WFM='CC',BF1='CC',AF1,J1/24,..

   Rewinds the device mechanism, writes one file mark command chained, backspaces one file mark command chained, advances one file mark, and repeats the write/backspace/advance file mark sequence until 25 file marks are written.

2. ..,BF25,AF25,:

   Then the test verifies the existence of the 25 file marks by backspacing 25 file marks and advancing 25 file marks.

3.16 Test 15--Write/Backspace/Advance File Marks (Variable Gap Length)

The test sequence follows:

1. RRW,PY=1,1,WFM1,ER,WFM1,BF2,AF2,PY+1,..

   Rewinds the device mechanism and defaults the erase interation count parameter Y to one, writes one file mark, performs Y erases, writes one file mark, backspaces two file marks, advances two file marks, increments the erase iteration count parameter Y by one, and repeats the write/erase/write file mark sequence until 10 pairs of file marks are written with each file mark pair separated by a variable length gap.

2. ..,J1/9,..

   The result is file marks one and two are separated by a 3.5-inch gap, file marks three and four are separated by a 7-inch gap, file marks five and six are separated by a 10.5-inch gap, and file marks 19 and 20 are separated by a 3-foot gap.

3. ..,BF20,AF20,:

   The test then verifies the existence of the 20 file marks by backspacing 20 file marks and advancing 20 file marks. Parameter Y is set to one.

3.17 Test 16—File Mark Rejection

The test sequence follows:

1. RWW,PD='13131313',W4096,1,WFM1,RW,..

   Rewinds the device mechanism and loads the output buffer with a pattern known to produce a CRC and LRC character equivalent to that of a file mark, provides the record length is 4096 characters, writes one 4096-character record, writes one file mark, and rewinds the device mechanism.

2. ..,R4096,RW,AR1,:

   The test then reads one 4096-character record and verifies that the High-Speed Tape Processor does not report End-of-File (EOF) status on the record read, rewinds the device, and advances one record.

3.18 Test 17—Write/Backspace/Write Record

The test sequence follows:

1. RWW,PD='3C3C3C3C',PI=0,1,PI+1,W4096,BR1,W4096,J1/24,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, writes one 4096-byte record, backspaces one record, writes one 4096-byte record, and repeats the write/backspace/write sequence until 25 records are written.

2. ..,WFM1,BF1,BR25,AR25,AF1,:

   The test then writes one file mark, backspaces one file mark, backspaces 25 records, advances 25 records, and advances one file mark.

3.19 Test 18—Write/Backspace/Advance Record

The test sequence follows:

1. RWW,PD='C3C3C3C3',PI=0,1,PI+1,W4096,BR1,AR1,J1/24,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, writes one 4096-byte record, backspaces one record, advances one record and repeats the write/backspace/advance sequence until 25 records are written.

2. ..,WFM1,BF1,BR25,AR25,AF1,:

   The test then writes one file mark, backspaces one file mark, backspaces 25 records, advances 25 records, and advances one file mark.

3.20 Test 19--Write/Backspace/Read Record

The test sequence follows:

1. RWW,PD='AAAA5555',PI=0,1,PI+1,W512,BR1,R512,J1/24,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, writes one 512-byte record, backspaces one record, reads one 512-byte record, and repeats the write/backspace/read sequence until 25 records are written.

2. ..,WFM1,BF1,BR25,AR25,AF1,:

   The test then writes one file mark, backspaces one file mark, backspaces 25 records, advances 25 records, and advances one file mark.

3.21 Test 20--Write/Rewind/Read Record

The test sequence follows:

1. RWW,PD='5555AAAA',PI=0,1,PI+1,W1024,J1/24,RW,PI=0,2,PI+1,R1024,J2/24,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, and writes 25 1024-byte records, rewinds the device mechanism, reinitializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, and reads 25 1024-byte records.

2. ..,WFM1,BF1,BR25,AR25,AF1,:

   The test then writes one file mark (..,WFM1,..), backspaces one file mark, backspaces 25 records, advances 25 records, and advances one file mark.

3.22 Test 21--Write/Rewind/Read Record (Variable Gap Length)

The test sequence follows:

1. RWW,PD='A5A5A5A5',PI=0,PY=1,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, and initializes the erase iteration counter parameter Y to one.

2. ..,1,PI+1,W2048,ER,PI+1,W2048,..

   Loads the first word of the output buffer with the record ID, writes one 2048-byte record, performs parameter Y erases, increments the record ID, and writes one 2048-byte record.

3. ..,BR2,AR2,PY+1,J1/9,..

   The test then backspaces two records, advances two records, then increments the erase iteration counter parameter Y by one and repeats the above sequence until 10 pairs of records are written, with each record pair separated by a variable length gap. Records one and two are separated by a 3.5-inch gap, records three and four are separated by a 7-inch gap, records five and six are separated by a 3-foot gap.

4. ..,RW,PI=0,2,PI+1,R2048,J2/19,..

   The test then rewinds the device mechanism, reinitializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, reads one 2048-byte record, and repeats the above sequence until 20 records are read.

3.23 Test 22—Write/Backspace/Read Record (Variable Gap Length)

The test sequence follows:

1. RWW,PD='5A5A5A5A',PI=0,PY=1,..

   Rewinds the device mechanism and loads the output buffer with a fixed data pattern, initializes the record ID parameter I to zero, and initializes the erase iteration counter parameter Y to one.

2. ..,1,PI+1,W4096,ER,PI+1,..

   Loads the first word of the output buffer with the record ID, writes one 4096-byte record, performs parameter Y erases, and increments the record ID.

3. ..,W2048,BR2,PI-1,R4096,PI+1,R2048,PY+1,..

   Writes one 2048-byte record, backspaces two records, decrements the record ID by one, reads one 4096-byte record, increments the record ID parameter I by one, reads one 2048-byte record, and increments the erase iteration counter parameter Y by one.

4. ..,J1/9,..

   Repeats the above sequence until 10 pairs of records are written with each record pair separated by a variable length gap. Records one and two are separated by a 3.5-inch gap, records three and four are separated by a 7-inch gap, records five and six are separated by a 10.5-inch gap, and records 10 and 20 are separated by a 3-foot gap.

5. ..,WFM1,BF1,BR20,PI=0,2,PI+1,R4096,PI+1,R2048,J2/9,AF1,:

   The test then writes one file mark, backspaces one file mark, backspaces 20 records, initializes the record ID parameter I to zero, loads the first word of the output buffer with the record ID, reads one 4096-byte record, increments the record ID parameter by one, reads one 2048-byte record, repeats the above sequence until 10 pairs of records are read, and then advances one file mark.

3.24 Test 23--Write/Rewind/Read Record (Fixed Length/Fixed Data)

The test sequence follows:

1. RWW,PW=4096,PR=25,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096-bytes per record, and defaults the record count parameter R to 25 records per block.

2. ..,1,LO,WR,LZ,WR,..

   Loads the output buffer with an all ones data pattern and writes 25 records of parameter W bytes, loads the output buffer with an all zeros data pattern, and writes 25 records of parameter W bytes.

3. ..,LA,WR,..

   Loads the output buffer with the 64-bytes ASCII printable subset and writes 25 records of parameter W bytes.

4. ..,LBP,WR,..

   Loads the output buffer with a binary progression data pattern and writes 25 records of parameter W bytes.

5. ..,LFO,WR,..

   Loads the output buffer with a floating one data pattern and writes 25 records of parameter W bytes.

6. ..,LFZ,WR,..

   Loads the output buffer with a floating zero data pattern and writes 25 records of parameter W bytes.

7. ..,PW/2,J1/9,..

   The test then divides the transfer count parameter W by two and repeats the above sequence until 25 records of each data pattern are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 incremental steps.

8. ..,RWW,PW=4096,2,LO,RR,LZ,RR,LA,RR,LBP,RR,LFO,RR,LFZ,RR,PW/2,J2/9,:

   Rewinds the device mechanism and reads the entire sequence.

9. Parameter R is set to one.

3.25 Test 24—Write/Backspace/Read Record (Variable Length/Fixed Data)

The test sequence follows:

1. RWW,PW=4096,PR=1,PI=0,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record, defaults the record count parameter R to one record per block, and initializes the record ID parameter I to zero.

2. ..,1,LNR,PI+1,W='CC',BR='CC',..

   Loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, writes one record of parameter W bytes command chained, and backspaces a record command chained.

3. ..,RR,J1/20,PW/2,J1/9,..

   The test then reads one record of parameter W bytes, repeats the write/backspace/read sequence until 21 records of parameter W bytes have been written, divides the transfer count parameter W by two and repeats the above sequence until 200 records of random data are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 increment steps.

4. ..,2,PW*2,3,LNR,PI+1,W='CC',BR='CC',RR,J3/20,J2/9,:

   The test then multiplies the transfer count parameter W by two and repeats the write/backspace/read sequence until 200 records of random data are written with transfer counts varied from eight bytes per record to 4096 bytes per record in 10 incremental steps.

3.26 Test 25—Write/Backspace/Read (Blocked Files)

The test sequence follows:

1. RWW,PW=4096,PI=0,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record, and initializes the record ID parameter I to zero.

2. ..,LIR,1,WFM1,2,LNR,PI+1,WR1,J2/19,BR1,RR1,BR10,AR9,RR1,BF1,AF1,AR19,RR1,..

   Loads the output buffer with a random data pattern, writes one file mark, loads the output buffer with a new random data pattern, loads the first word of the output buffer with the record ID, writes 20 records of parameter W bytes, backspaces one record, reads one record of parameter W bytes, backspaces 10 records, advances nine records, reads one record of parameter W bytes, backspaces one file mark, advances one file mark, advances 19 records, and reads one record of parameter W bytes.

3. ..,J1/1,PW/2,J1/9,..

   The test repeats the entire sequence twice, then divides the transfer count parameter W by two and repeats the above sequence until 20 files (each file contains 20 records of random data) are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 incremental steps.

4. ..,RW,PW=4096,PI=0,LIR,3,AF1,4,LNR,PI+1,RR1,J4/19,J3/1,..

   The test then rewinds the device mechanism and defaults the transfer count parameter W to 4096 bytes per record, initializes the record ID parameter I to zero and loads the output buffer with random data, advances one file mark, loads the output buffer with a new random data pattern, loads the first word of the output buffer with the record ID, reads 20 records of parameter W bytes, and repeats the advance file mark/read sequence twice.

5. ..,PW/2,J3/9,RW,AF20,AR19,PW*2,RR1,:

   The test then divides the transfer count parameter W by two and repeats the above sequence until 20 files (each file contains 20 records of random data) have been read with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 incremental steps, rewinds the device mechanism, advances 20 file marks, advances 19 records, multiplies parameter W bytes by 2, and reads one record.

3.27 Test 26—Write Record (Random Length)

The test sequence follows:

1. RWW,PW=V,PB=319,..

   Rewinds the device mechanism and defaults the transfer count parameter W to the length of the diagnostic's output buffer and initializes the random number generator parameter B to contain an odd integer value.

2. ..,PI=0,LIR,1,LNR,PI+1,PB*61,PW=B,PW.A.'FFF,..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, calculates a 12-bit random number, and inserts it into the transfer count parameter W.

3. ..,WR1,J1/1999,:

   Writes one record of parameter W bytes and repeats the above sequence until 2000 random length records are written.

The tape produced by this test may be removed and maintained for device mechanism compatibility testing. To test for device mechanism compatibility, mount this tape on the device mechanism in question and execute Test 27 only.

3.28 Test 27—Read Record (Random Length)

Test 26 is a prerequisite for this test. The test sequence follows:

1. RWW,PW=V,PB=319,..

   Rewinds the device mechanism and defaults the transfer count parameter W to the length of the diagnostic's output buffer and initializes the random number generator parameter B to contain an odd integer value.

2. ..,PI=0,LIR,1,LNR,PI+1,PB*61,PW=B,PW.A.'FFF,..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, calculates a 12-bit random number, and inserts it into the transfer count parameter W.

3. ..,RR1,J1/1999,:

   Reads one record of parameter W bytes and repeats the above sequence until 2000 random length records are read.

3.29 Test 28—Write/Read Backward Record (Variable Length/Fixed Data)

The test sequence follows:

1. RWW,PW=4096,PR=1,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record, and defaults the record count parameter R to one.

2. ..,PI=0,1,LNR,PI+1,W='CC',RB,AR1,J1/20,..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, writes one record of parameter W bytes command chained, reads backward one record of parameter W bytes, advances one record, and repeats the write/read backward/advance record sequence until 21 records of parameter W bytes are written.

3. ..,PW/2,J1/9,..

   The test then divides the transfer count parameter W by two and repeats the above sequence until 200 records of random data are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 increment steps.

4. ..,2,PW*2,3,LNR,PI+1,W='CC',RB,AR1,J3/20,J2/9,:

   Then, multiplies the transfer count parameter W by two and repeats the write/read backward/advance record sequence until 21 records of random data are written with transfer counts varied from eight bytes per record to 4096 bytes per record in 10 incremental steps.

3.30 Test 29--Write/Backspace/Read Compare (Variable Length/Fixed Data)

The test sequence follows:

1. RWW,PW=4096,PR=1,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record, and defaults the record count parameter R to one.

2. ..,PI=0,1,LRN,PI+1,W='CC',BR='CC',..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, writes one record of parameter W bytes command chained, and backspaces one record command chained.

3. ..,RC,J1/20,..

   The test then reads and compares one record of parameter W bytes and repeats the write/backspace/read/compare sequence until 21 records of parameter W bytes are written.

4. ..,PW/2,J1/9,..

   The test then divides the transfer count parameter W by two and repeats the above sequence until 200 records of random data are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 increment steps.

5. ..,2,PW*2,3,LNR,PI+1,W='CC',BR='CC',RC,J3/20,J2/9,:

   The test then multiplies the transfer count parameter W by two and repeats the write/backspace/read compare sequence until 200 records of random data are written with transfer counts varied from 10 bytes per record to 4096 bytes per record in 10 increment steps.

3.31 Test 30--Two Data Chain Read

The test sequence follows:

1. RWW,PW=4096,PR=1,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record and defaults the record count parameter R to one.

2. ..,PI=0,1,LNR,PI+1,LIZ,W='CC',BR='CC',PB=W,PW/2,R='DC',RR,PW=B,J1/20,..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, loads the input buffer with zeros, writes one record of parameter W bytes command chained, backspaces a record command chained, sets parameter B equal to parameter W, divides parameter W by two, reads data chained to another read command of parameter W bytes per records, restores the value of parameter W prior to the read data chained command, and repeats the write/back-space/read data chained sequence until 200 records of parameter W bytes are written.

3. ..,PW/2,J1/7,2,PW*2,3,LNR,PI+1,PB=W,LIZ,W='CC',BR='CC',PW/2,R='DC',RR, PW=B,J3/20,J2/7,:

   The test then divides the transfer count parameter W by two and repeats the above sequence until 200 records of random data are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 increment steps, multiplies the transfer count parameter W by two, and repeats the write/backspace/read sequence until 200 records of random data are written with transfer counts varied from 10 bytes per record to 4096 bytes per record in 17 incremental steps.

3.32 Test 31—Four Data Chain Read

This test sets up a four data chained Read IOCL queue with a starting byte count of 64 in the first two IOCDs and a starting byte counts of 63 and 62 in the next two IOCDs. The byte count in each of these IOCDs is decremented by one until the byte count is equal to two. When all four IOCDs have a byte count of two, the test is completed.

The test sequence follows:

1. RWW,PB=64,PC=63,PF=62,1,PW=0,PW+B,PW+B,PW+C,PW+F,..

   Rewinds the device mechaism, defaults the first and second read data chain command parameters to 64 bytes, defaults the third data chain command to 63 bytes and defaults the last read command to 62 bytes, then sets parameter W to zero, adds parameter B to W, adds parameter C to W, and adds parameter F to W to obtain the total number of bytes to be transferred by the Write command.

2. ..,LIZ,LBP,WR1,BR1,PW=B,R='DC',R='DC',PW=C,R='DC',PW=F,RR1,..

   Clears the input buffer of parameter W bytes, loads the output buffer with parameter W bytes of a binary progression, writes one record of W bytes, then backspaces a record, sets parameter W to B and inserts the first Read Data Chain command into the IOCL queue, inserts the second Read Data Chain command into the IOCL queue with the same parameter byte count, sets parameter W to C and inserts the third read data chain command into the IOCL queue, then sets parameter W to F, inserts a Read command into the IOCL queue, and executes the Start I/O instruction.

3. ..,J/4F=2,PB-1,PC-1,PF-1,J1,..

   Jumps to directive statement number four if parameter F is equal to two, otherwise the byte counts in parameters B, C, and F are decremented by one, and the program repeats the test by jumping to directive statement one.

4. Directive Statement Four Processing

   ..,4,J5/C=2,PC-1,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter C. If the minimum is reached, the diagnostic jumps to directive statement five; otherwise, the byte counts in parameters B and C are decremented by one and the program jumps to directive statement one.

5. Directive Statement Five Processing

   ..,5,J6/B=2,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter B. If the minimum is reached, the diagnostic jumps to directive statement number six; otherwise, the byte count in parameter B is decremented by one and the diagnostic jumps to directive statement one.

6. Directive Statement Six Processing

   ..,6,RWW,:

   The diagnostic rewinds the device and exits the test.

3.33 Test 32—Skip Read Data Chained

This test sets up three Data Chain Read commands in the IOCL queue with the skip flag set on the first command in the IOCL queue. The test starts with a byte of 128 and decrements the byte count by one byte until the byte count equals two.

The test sequence follows:

1. RWW,PB=V,PB/32,PC=B,1,PW=C,PW+B,..

   Rewinds the device mechanism, defaults the transfer count parameter B to parameter V or 4096 bytes, divides parameter B by 32, sets parameter C equal to parameter B, sets parameter W to C, which is equal to 516 bytes, adds the variable parameter B to parameter W, and clears the input buffer of parameter W bytes.

2. ..,LIZ,LBP,W,BR1,..

   Loads the output buffer with parameter W bytes of a binary progression and writes one record of W bytes followed by a backspace one record.

3. ..,PW=B,LZ,..

   The test then sets the parameter W equal to the variable byte count in parameter B and clears the output buffer of parameter B bytes, inserts a Skip Data Chain Read Command into the IOCL queue, sets parameter W equal to C, divides parameter W by two, and inserts a Data Chained Read Command into the IOCL.

4. ..,R='DC,SKIP',PW=C,PW/2,R='DC',RR,PB-1,J1/B=>2,:

   The test then inserts a Read command and issues a Start I/O instruction, decrements the variable byte count parameter B by one, and continues the write/backspace record/skip read data chain/read data chain/read/sequence until the variable byte count in parameter B is decremented to 10.

3.34 Test 33—Two Data Chain Write

The test sequence follows:

1. RWW,PW=4096,PR=1,..

   Rewinds the device mechanism, defaults the transfer count parameter W to 4096 bytes per record, and defaults the record count parameter R to one.

2. ..,PI=0,1,LRN,PI+1,PB=W,LIZ,PW/2,W='DC',W='CC,PW=B,BR='CC',..

   Initializes the record ID parameter I to zero, loads the output buffer with a random data pattern, loads the first word of the output buffer with the record ID, sets parameter B equal to parameter W and loads the input buffer with zeros, divides parameter W by two, and Writes Data Chained to another Write command of parameter W bytes which is command chained to a Backspace Record Command Chained.

3. ..,RR,J1/20,..

   The test then reads one record of parameter W bytes and repeats the write data chained/backspace/read sequence until 200 records of parameter W bytes are written.

4. ..,PW/2,J1/7,..

   The test then divides the transfer count parameter W by two and repeats the above sequence until 200 records of random data are written with transfer counts varied from 4096 bytes per record to 10 bytes per record in 10 increment steps.

5. ..,2,PW*2,3,LNR,PI+1,PB=W,LIZ,PW/2,W=DC,W='CC',PW=B,BR='CC',RR,J3/20,J2/7,:

   The test then multiplies the transfer count parameter W by two and repeats the write/backspace/read compare sequence until 200 records of random data are written with transfer counts varied from 10 bytes per record to 4096 bytes per record in seven incremental steps.

3.35 Test 34—Four Data Chain Write (Variable Length)

This test sets up a four data chained Write commands in the IOCL queue with a starting byte count of 64 in the first two IOCDs and a starting byte counts of 63 and 62 in the next two IOCDs. The byte count in each of these IOCDs is decremented by one until the byte count is equal to two. When the byte count in all four IOCDs is two, the test is completed.

The test sequence follows:

1. RWW,PB=64,PC=63,PF=62,1,PW=0,PW+B,PW+B,PW+C,PW+F,LIZ,PG=W,..

   Rewinds the device mechanism, defaults the first and second Write Data Chain command parameter B to 64 bytes, defaults the third Write Data Chain command parameter C to 63 bytes and defaults the last Write command parameter F to 62 bytes, then sets parameter W to zero, adds parameter B to W, adds parameter B to W, adds parameter C to W, adds parameter F to W, clears the input buffer, then sets parameter G to W, which totals the number of bytes used by the Read command.

2. ..,LBP,PW=B,W='DC',W='DC',PW=C,W='DC',PW=F,WR1,BR1,PW=G,RR1,..

   Loads the output buffer with parameter W bytes of a binary progression. Sets parameter W to B and inserts the first Data Chain Write command into the IOCL queue, inserts the second Write Data Chain command into the IOCL queue, sets parameter W to C and inserts the third command into the IOCL queue, sets parameter W to F bytes and inserts the last Write command into the IOCL queue, and executes the Start I/O instruction. The test then backspaces one record, sets parameter W to G, and reads one record.

3. ..,J/4F=2,PB-1,PC-1,PF-1,J1,..

   Jumps to directive statement four if parameter F is equal to two; otherwise, the byte counts in parameters B, C, and F are decremented by one and the program repeats the test by jumping to directive statement one.

4. Directive Statement Four Processing

   ..,4,J5/C=2,PC-1,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter C. If the minimum is reached, the diagnostic jumps to directive statement five; otherwise, the byte counts in parameters B and C are decremented by one and the program jumps to directive statement one.

5. Directive Statement Five Processing

   ..,5,J6/B=2,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter B. If the minimum is reached, the diagnostic jumps to directive statement six; otherwise, the byte count in parameter B is decremented by one and the diagnostic jumps to directive statement one.

6. Directive Statement Six Processing

   ..,6,RWW,:

   The diagnostic rewinds the device and exits the test.

3.36 Test 35—Read Compare Data Chained

This test inserts three Data Chain Read Compare commands with a byte count of 128 into the IOCL queue. The test then issues a Start I/O instruction and verifies that the read compare IOCL queue functions properly. The test then decrements the byte count by one in each IOCD and continues testing until a byte count of two is reached.

The test sequence follows:

1. RWW,PB=V,PB/32,PC=B,1,PW=B,PW*4,..

   Rewinds the device mechanism, defaults the transfer count variable parameter B to parameter V or 4096 bytes, divides parameter B by eight, then sets parameter C equal to parameter B, sets parameter W equal to parameter B, and multiplies parameter W by four.

2. ..,LIZ,LBP,W,BR1,PW=B,RC='DC',RC='DC',..

   Clears the input buffer by the byte count contained in parameter W, loads the output buffer with a binary progression of parameter W bytes, writes one record of W bytes followed by a Backspace Record command, sets parameter W equal to B, and inserts two Read Compare Data Chained commands into the IOCL queue.

3. ..,RC,PB-1,J1/B=>2,RWW,:

   The test then inserts a Read Compare command into the IOCL queue, causes the execution of a Start I/O instruction of the commands in the IOCL queue, decrements the variable byte count in parameter B by one and continues the write/backspace record/read compare data chain/read compare data chain/read compare sequence until the byte count in parameter B is decremented to two. The diagnostic rewinds the device and exits the test.

3.37 Test 36—Four Data Chain Read Compare

This test sets up a four data chained Read IOCL queue with a starting byte count of 64 in the first two IOCDs and a starting byte count of 63 and 62 in the next two IOCDs. The byte count in each of these IOCDs is decremented by one until the byte count is equal to two. When the byte count in all four IOCDs is two, the test is completed.

The test sequence follows:

1. RWW,PB=64,PC=63,PF=62,1,PW=0,PW+B,PW+B,PW+C,PW+F,..

   Rewinds the device mechanism, defaults the first and second Read Compare Data Chain command parameters to 64 bytes, defaults the third Data Chain command to 63 bytes and defaults the last Read Compare command to 62 bytes, then sets parameter W to zero, adds parameter B to W, adds parameter C to W, and adds parameter F to W to obtain the total number of bytes to be transferred by the Write command.

2. ..,LIZ,LBP,WR1,BR1,PW=B,RC='DC',RC='DC',PW=C,RC='DC',PW=F,RC,..

   Clears the input buffer of parameter W bytes, loads the output buffer with parameter W bytes of a binary progression, writes one record of W bytes, then backspaces a record. Sets parameter W to B and inserts the first Read Compare Data Chain command into the IOCL queue, inserts the second Read Compare Data Chain command into the IOCL queue with the same parameter byte count, sets parameter W to C and inserts the third Data Chain command into the IOCL queue, then sets parameter W to F, inserts a Read Compare command into the IOCL queue, and causes the Start I/O Instruction to be executed.

3. ..,J4/F=2,PB-1,PC-1,PF-1,J1,..

   Jumps to directive statement four if parameter F is equal to two; otherwise, the byte counts in parameters B, C, and F are decremented by one and the program repeats the test by jumping to directive statement one.

4. Directive Statement Four Processing

   ..,4,J5/C=2,PC-1,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter C. If the minimum is reached, the diagnostic jumps to directive statement five; otherwise, the byte counts in parameters B and C are decremented by one and the program jumps to directive statement one.

5. Directive Statement Five Processing

   ..,5,J6/B=2,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter B. If the minimum is reached, the diagnostic jumps to directive statement six; otherwise, the byte count in parameter B is decremented by one and the diagnostic jumps to directive statement one.

6. Directive Statement Six Processing

   ..,6,RWW,:

   The diagnostic rewinds the device and exits the test.

3.38 Test 37—Skip Read Compare Data Chained

The test sequence follows:

1. RWW,PB=V,PB/32,1,PW=B,PW*4,LIZ,LBP,..

   Rewinds the device mechanism, defaults the variable transfer count parameter B to V, divides parameter B by 32 and then sets parameter W to the variable parameter B, multiplies parameter W by four, clears the input buffer, and sets the output buffer to parameter W bytes.

2. ..,W,BR1,PW=B,LZ,..

   Performs a Write command of parameter W bytes followed by a Backspace Record command, sets parameter W to the variable parameter B, and clears the output buffer of parameter B bytes.

3. ..,RC='DC,SKIP',RC='DC',RC='DC',RC,..

   Inserts a Skip Data Chained Read Compare command into the IOCL queue, inserts two Data Chained Read compare commands into the IOCL queue followed by a Read Compare command, and issues a Start I/O instruction.

4. ..,PB-1,J1/B=>2,RWW,:

   Decrements the variable byte count parameter B by one and continues the write/backspace record/skip data chain read compare/data chain read compare/read compare sequence until the variable byte count in parameter B is decremented to 10. The diagnostic then rewinds the device and exits the test.

3.39 Test 38—Four Data Chain Skip in Second and Fourth Read Compare

The test sets up a four data chained Read Compare IOCL queue with a skip command flag set in the second and fourth Read Compare IOCD. The starting byte count of the first two IOCDs is 64. The byte count in the next two IOCDs is 63 and 62, respectively. The byte count in each IOCD is decremented by one until the byte count is equal to two. When the byte count in all four IOCDs is equal to two, the test is completed.

The test sequence follows:

1. RWW,PB=64,PC=63,PF=62,1,PW=0,PW+B,PW+B,PW+C,PW+F,..

   The test rewinds the device, defaults the first and second Read Compare Data Chained parameters to 64 bytes, defaults the third data chain byte count to 63 and defaults the last Read Compare byte count to 62 bytes, then sets parameter W to zero, adds the four data chained byte counts to obtain the total number of bytes by adding parameter B to parameter W, parameter B to parameter W, parameter C to parameter W, and parameter F to parameter W, which is the total number of bytes used to clear the input buffer.

2. ..,LIZ,PG=W,LZ,PW-F,LBP,PW-C,LZ,PW=B,LBP,PW=G,..

   Clears the input buffer of parameter W bytes, saves the value of parameter W in parameter G and loads zeros to the entire input buffer, subtracts parameter F from parameter W and sets the output buffer to a binary progression, subtracts parameter C from parameter W and loads the output buffer with zeros, sets parameter W to parameter B, and sets a binary progression to the output buffer, then prepares parameter W for the Write command by setting parameter W to parameter G.

3. ..,WR1,BR1,PW=B,RC='DC',RC='DC',SKIP,PW=C,RC='DC',PW=F,RC='SKIP',..

   Writes one record of parameter W bytes, then backspaces a record, sets parameter W to parameter B, which is the byte count for the first two Read Compare commands, then inserts the first Read Compare Data Chain command into the IOCL queue, inserts the second Read Compare Data Chain command with the Skip command flag set into the IOCL queue, sets the next parameter C byte count to W and inserts the third command into the IOCL queue then sets parameter W to the last byte count parameter F and inserts the Read Compare command with the Skip command flag set into the IOCL queue, and causes the Start I/O instruction to be executed.

4. ..,J/4F=2,PB-1,PC-1,PF-1,J1,..

   Jumps to directive statement four if parameter F is equal to two; otherwise, the byte counts in parameters B, C, and F are decremented by one and the program repeats the test by jumping to directive statement one.

5. Directive Statement Four Processing

   ..,4,J5/C=2,PC-1,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter C. If the minimum is reached, the diagnostic jumps to directive statement five; otherwise, the byte counts in parameters B and C are decremented by one and the program jumps to directive statement one.

6. Directive Statement Five Processing

   ..,5,J6/B=2,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of two is reached by parameter B. If the minimum is reached, the diagnostic jumps to directive statement six; otherwise, the byte count in parameter B is decremented by one and the diagnostic jumps to directive statement one.

7. Directive Statement Six Processing

   ..,6,RWW,:

   The diagnostic then rewinds the device and exits the test.

3.40 Test 39—Read Backward Data Chained

The test sequence follows:

1. RWW,PB=V,PB/32,PC=B,1,PW=B,PW*2,..

   Rewinds the device mechanism, defaults the transfer count parameter B to parameter V or 4096 bytes, divides parameter B by 32, sets parameter C equal to parameter B, sets parameter W to parameter B, and multiplies parameter W by 2.

2. ..,LIZ,LBP,WR,LBP,PW=B,RB='DC',..

   Clears the input buffer of parameter W bytes, loads the output buffer with parameter W bytes of a binary progression, writes one record of W bytes, sets parameter W equal to parameter B, and inserts a Data Chained Read Backward command into the IOCL queue.

3. ..,RB,..

   Inserts a Read Backward command into the IOCL queue and causes an SIO instruction to execute the commands in the IOCL queue.

4. ..,AR1,PB-1,J1/B=>2,RWW,:

   Advances one record, then decrements the variable parameter B by one and continues the write/read backward/advance record sequence until the variable byte count in parameter B is decremented to two. The diagnostic then rewinds the device and exits the test.

3.41 Test 40—Four Data Chained Read Backward

The test sets up a four Data Chained Read Backward IOCL queue with a starting byte count of 64 in the first two IOCDs and a byte count of 63 and 62 in the next two IOCDs. The byte count in each of these two IOCDs is decremented by one until the byte count reaches two. When the byte count in all four IOCDs is equal to two, the test is completed.

The test sequence follows:

1. RWW,PB=64,PC=63,PF=62,1,PW=0,PW+B,PW+B,PW+C,PW+F,..

   Rewinds the device mechanism, defaults the first and second Read Backward Data Chain command to 64 bytes, defaults the third Data Chain command to 63 bytes and defaults the last Read Backward command to 62 bytes, then sets parameter W to zero, adds parameter B to parameter W, adds parameter B to parameter W, adds parameter C to parameter W, and adds parameter F to parameter W to obtain the total number of bytes to be transferred by the Write command.

2. ..,LIZ,LBP,WR1,PW=B,RB='DC',RB='DC',PW=C,RB='DC',PW=F,RB,AR1,..

   Clears the input buffer of parameter W bytes, loads the output buffer with parameter W bytes of a binary progression, writes one record of W bytes, sets parameter W to parameter B, inserts the first Read Backward Data Chain command into the IOCL queue, inserts the second Read Backward Data Chain command into the IOCL queue with the same parameter byte count, sets parameter W to parameter C, inserts the third Read Backward Data Chain command into the IOCL queue, then sets parameter W to parameter F, inserts a Read Backward command into the IOCL queue, causes the Start I/O Instruction to be executed, and then advances one record.

3. ..,J4/F<12,PB-1,PC-1,PF-1,J1,..

   Jumps to directive statement four if parameter F is equal to 12; otherwise, the byte counts in parameters B, C, and F are decremented by one and the program repeats the test by jumping to directive statement one.

4. Directive Statement Four Processing

   ..,4,J5/C<12,PC-1,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of 12 is reached by parameter C. If the minimum is reached, the diagnostic jumps to directive statement five; otherwise, the byte counts in parameter B and C are decremented by one and the program jumps to directive statement one.

5. Directive Statement Five Processing

   ..,5,J6/B<12,PB-1,J1,..

   The diagnostic checks to see if the minimum byte count of 12 is reached by parameter B. If the minimum is reached, the diagnostic jumps to directive statement six; otherwise, the byte count in parameter B is decremented by one and the diagnostic jumps to directive statement one.

6. Directive Statement Six Processing

   ..,6,RWW,:

   The diagnostic then rewinds the device and exits the test.

3.42 Test 41--Read Backward Skip Data Chained

The test sequence follows:

1. RWW,PB=V,PB/32,PC=B,1,PW=B,PW*4,..

   Rewinds the device mechanism, defaults the transfer count parameter B to parameter V or 4096 bytes, divides parameter B by thirty two, sets parameter C equal to parameter B and sets parameter W to parameter B, and multiplies parameter W by four.

2. ..,LIZ,LBP,W,LZ,PW=B,..

   Clears the input buffer of parameter W bytes, loads the output buffer with parameter W bytes of a binary progression, writes one record of parameter W bytes, clears the output buffer, then sets parameter W to the variable byte count in parameter B.

3. ..,RB='DC,SKIP',PW=B,PW*3,LBP,PW=B,..

   Inserts a Read Backward Data Chain, Skip command into the IOCL, sets parameter W equal to parameter B, multiplies parameter W by three, loads the output buffer with parameter W bytes, and then resets parameter W to the decrementing byte count in parameter B.

4. ..,RB='DC',RB='DC',RB,AR1,PB-1,J1/B=>16,RWW,:

   Inserts a Read Backward Data Chain command of parameter W bytes into the IOCL queue inserts a second Read Backward Data Chain command into the IOCL queue with the same parameter byte count, inserts a Read Backward command into the IOCL queue, causes the Start I/O instruction to be executed, advances a record, decrements the variable parameter B by one and continues the write/skip data chain read backward/data chain read backward/read backward advance a read sequence until the variable byte count in parameter B is decremented to 16. The diagnostic then rewinds the device and exits the test.

3.43 Test 42—Variable Buffer Starting Address

This test increments the starting input and output address by one byte up to a maximum of eight to verify that the Write, Read, Read Backward, and Read Compare commands function properly. Each time the address is incremented by one byte, the data transfer count is set to 12 and is decremented down to two bytes. The test is completed when the address is equal to the starting address plus eight bytes of both input and output buffers and the byte count is decremented to two.

The test sequence follows:

1. RWW,PG=0,PW=300,LBR,..

   The test rewinds the device mechanism and sets parameter G to zero (parameter G is used by the BAC directive to offset the starting address of the output buffers), sets parameter W to 300, and loads the output buffer with parameter W bytes of a binary regression.

2. ..,1,PB=12,BAC,..

   Sets parameter B to 12 and sets the buffer addresses to the original buffer address modified by the contents of parameter G, using the ..,BAC,.. directive.

3. ..,2,PW=B,WR1,RRB1,RR1,BR1,RC,..

   Sets parameter W to parameter B and writes one record, performs a Read/Backward of one record, performs a Read command of one record, then backspaces a record and performs a Read Compare.

4. ..,PB-1,J2/B>1,..

   Decrements parameter B by one, which is used as the new byte count for the Write/Read commands, and continues to directive statement two until parameter B is less than two.

5. ..,PG+1,J1/8,..

   Adds one to parameter G, which increments the starting address of the I/O buffers by one byte if the ..,BAC,.. directive is performed, and continues to directive statement one 8 times.

6. ..,PG=0,PN=0,BAC,RWW,:

   Sets parameters G and N to zero, adds parameter G to the starting addresses of the I/O buffers and concludes the test with a rewind of the device.

3.44 Test 43—Post Program Control Interrupt

This test verifies that an interrupt is generated if a command, which has the Post Program Control Interrupt bit set, is executed. The received status is checked for Post Program Control Interrupt set with no Device End or Channel End.

The test also verifies that a subsequent interrupt is received which posts a status of Channel End and Device End. This test is composed of five subtests.

3.44.1 Subtest A

W='CC,PPCI',W='CC',W='NSIO',";'

This subtest inserts a Write Command Chained with the Post Program Control Interrupt bit set, followed by a Write Command Chained to a Write command into the IOCL queue.

The test then issues a Start I/O (SIO) instruction and verifies that an interrupt is received after the first Write Command Chained is executed and that another interrupt is received after the last Write command is executed which posts a status of Channel End and Device End.

3.44.2 Subtest B

W='CC',W='CC,PPCI',W='NSIO'.";'

This subtest inserts a Write Command Chained to another Write Command Chained which has the Post Program Control Interrupt bit set, followed by another Write command into the IOCL queue.

The test then issues a Start I/O (SIO) instruction and verifies that an interrupt is received after the second Write Command Chained is executed and that another interrupt is received after the last Write command is executed which posts a status of Channel End and Device End.

3.44.3 Subtest C

WR3,BR3

This subtest initially writes three records and then backspaces three records.

R='CC,PPCI',R='CC',R='NSIO',":'

A Read Command Chained with the Post Program Control Interrupt bit set, followed by a Read Command Chained to another Read command is inserted into the IOCL queue.

The test then issues a Start I/O (SIO) instruction and verifies that an interrupt is received after the first Read command which indicates that the status is Post Program Control Interrupt, followed by another interrupt after the third command which posts a status of Channel End and Device End.

3.44.4 Subtest D

WR3,BR3,..

This subtest initially writes three records and then backspaces three records.

..,R='CC',R='CC,PPCI',R='NSIO',":'

A Read Command Chained to a Read Command Chained command with the Post Program Control Interrupt test to another Read command is inserted into the IOCL queue.

The test then issues a Start I/O (SIO) instruction and verifies that an interrupt is received after the second Read command followed by another interrupt after the last Read command which posts a status of Channel End and Device End.

3.44.5 Subtest E

WR1,BR='CC',PW/3,R='DC',R='DC',R='CC,PPCI',BR='CC',BR='NSIO',";'

This subtest initially writes a record of parameter W bytes, then inserts a Backspace Record Command Chained command into the IOCL queue, divides the byte count in parameter W by three, inserts a Read Data Chained into the IOCL queue, inserts a Read Data Chained command into the IOCL queue followed by a Read Command Chained with the Post Program Control Interrupt bit set, then inserts a Backspace Record Command Chained to another Backspace Record into the IOCL queue.

The test then issues a Start I/O (SIO) instruction and verifies that an interrupt is received after the Read command with the PPCI bit set is executed. The program also verifies that another interrupt is received at the completion of all the commands in the IOCL queue. The final status should be Channel End and Device End.

3.45 Test 44--Halt I/O, Stop I/O, Test I/O Dynamic State

This test verifies that the Halt I/O, Stop I/O and Test I/O instructions perform properly if the device is in a dynamic state. The test also verifies that a Stop I/O instruction terminates if a Write command is data chained and a Stop I/O is performed. This test is composed of six subtests.

3.45.1 Subtest A

This test verifies that a Halt I/O instruction generates an interrupt while the device is executing an IOCL queue of five Write Command Chained commands. The test verifies that the Halt I/O interrupt is received before the end of the command chains are reached.

3.45.2 Subtest B

This test verifies that a Stop I/O instruction generates an interrupt while the device is executing an IOCL queue of five Write Command Chained commands. The test verifies that the Stop I/O interrupt is received before the end of the Write Command Chained commands is reached.

3.45.3 Subtest C

This test verifies that a Test I/O instruction generates a condition code setting of Channel Busy (1) when the device is beginning to execute an IOCL queue of five Write Command Chained commands. No interrupt is expected for the Test I/O instruction.

3.45.4 Subtest D

This test verifies that a Stop I/O instruction generates an interrupt while the device is executing an IOCL queue of five Write Data Chained commands. The subtest is designed to verify that the interrupt for the Stop I/O occurred after the first data chained command. The data is verified using a Read command to insure that the Write command terminated at the first Write Data Chained command.

3.45.5 Subtest E

This test verifies that a Halt I/O instruction generates an interrupt while the device is executing an IOCL queue of Write Data Chained commands. The test verifies that the Halt I/O interrupt is received before the end of the first Write Data Chained byte count reaches zero.

3.45.6 Subtest F

This test verifies that a Test I/O instruction generates a condition code setting of Subchannel Busy (CC=3) if a Test I/O instruction is issued while the device is rewinding.

3.46 Test 45—Scatter/Gather Write

This test verifies the Scatter/Gather Write capability of a device. The test begins with a scatter/gather bias of four and increments the bias by four until the total bias used by the Write command reaches 56. The test uses a binary progression and recreates the output buffer to appear the same as the data which is input via a Read command.

The test sequence follows:

1. RWW,PW=V,PW/2,PL=W,BCO,LBP,..

   Rewinds that device mechanism, sets parameter W to parameter V, then divides by two, sets parameter L to parameter W and changes the starting address of the output buffer by using the ..,BCO,.. directive (which modifies the starting address of the output buffer by adding the data value of parameter L to the original buffer starting address), and then loads the top half of the output buffer with parameter W bytes of a binary progression.

2. ..,PL=0,BCO,LBP,PL=W,PG=0,BAC,..

   Sets parameter L to zero, then changes the starting address of the output buffer by using the BCO directive to the original starting address and loads the first (bottom) half of the output buffer with parameter W bytes of a binary progression, sets parameter L to parameter W, sets parameter G to zero, and restores the starting addresses of both the input and output buffers to their original values by using the ..,BAC,.. directive.

3. ..,PO=4,PC=32,1,PW=C,SM='SGM,SGBC',..

   Sets parameter O to four, sets parameter C to 32 (this is the decrementing byte count), sets parameter W to parameter C and sets the mode register modified by the string arguments to set the Scatter/Gather mode bit 7 and the bias in parameter O into the appropriate two bytes in the mode register data.

4. ..,WR1,LIZ,LSG,BR='CC',SM='RSGM',..

   Performs the Scatter/Gather write command, clears the input buffer of parameter W bytes, then conditions the output buffer as it should appear after data is input by using the LSG directive (refer to the example at the end of this description), inserts a Backspace Record Command Chained into the IOCL queue, then inserts a Set Mode Register command into the IOCL queue, causes the Start I/O instruction to be executed, and modifies the Set Mode command by the string directive RSGM, which resets bit 7 in the mode register.

5. ..,RR1,PO+4,PW=L,LBP,PW=C,J1/O<60,..

   Performs a Read command and compares the data input to the data in the output buffer, which was prepared by using the LSG directive, then adds four to parameter O for the next bias, sets parameter W to parameter L and resets the output buffer to the original data pattern, sets parameter W to parameter C, and jumps to directive statement one until the bias data in parameter O is equal to 60.

6. ..,PO=4,PC-1,PW=C,J1/C=>8,PG=0,PL=0,BAC,SM='RSGM',RWW,:

   Sets the Scatter/Gather bias byte count in parameter O to four, decrements the byte count contained in parameter C by one, then sets the byte count in parameter W to C. This program then jumps to directive statement one until the byte count in parameter C is equal to or greater than eight; otherwise, the program sets parameters G and L to zero, restores the original starting addresses of the input and output buffers by use of the ..,BAC,.. directive, performs a Set Mode command, which resets the mode register bit 7 by using the string argument RSGM, and concludes by rewinding the device.

Example

The following example shows how the buffers appear before and after the Scatter/Gather mode is used.

- Scatter/Gather Bias is eight
- Output buffer before Write command

  00010203 04050607 08090A0B 0C0D0E0F 10111213

- Output buffer after Write command and using ..,LSG,.. directive

  00010203080 90A0B10111213

- Input buffer should be equal to the data in the output buffer

3.47 Test 46—Scatter/Gather Read

This test verifies the Scatter/Gather Read capability is functioning properly. The test begins with a Scatter/Gather bias of four and increments this bias until a maximum bias of 56 is reached. The data in the output buffer is manipulated to reflect the expected input data after the Read command is issued.

The test sequence follows:

1. RWW,PW=V,PW/2,PL=W,BCO,LBP,..

   Rewinds the device mechanism, sets parameter W to parameter V then divides by two, sets parameter L to parameter W and changes the starting address of the output buffer by using the ..,BCO,.. directive (which modifies the starting address of the output buffer by adding the data value of parameter L to the original buffer starting address), and then loads the top half of the output buffer with parameter W bytes of a binary progression.

2. ..,PL=0,BCO,LBP,PL=W,PG=0,BAC,..

   Sets parameter L to zero, then changes the starting address of the output buffer by using the BCO directive to the original starting address and loads the first (bottom) half of the output buffer with parameter W bytes of a binary progression, sets parameter L to parameter W, sets parameter G to zero, and restores the starting addresses of both the input and output buffers to their original values by using the ..,BAC,.. directive.

3. ..,PO=4,PC=32,1,PW=C,SM='RSGM',..

   Sets parameter O to four, sets parameter C to 32 (which is the decrementing byte count), sets parameter W to parameter C, sets the mode register modified by the string argument which resets the Scatter/Gather mode register bit 7 into the IOCL queue, and executes the Set Mode Register command.

4. ..,WR1,LIZ,SM='SGM,SGBC',..

   Performs the Write command, clears the input buffer of parameter W bytes, sets the mode register to set the Scatter/Gather mode of operation (bit 7) and the bias in parameter O into the appropriate two bytes in the mode register data.

5. ..,LSC,PF=C,COM1,..

   Sets the output buffer to the correct data pattern (which is expected because of the variable bias in the mode register) by using the ..,LSC,.. directive, sets parameter F to parameter C, then calculates the total number of bytes including the bias, and sets parameter W to that value.

6. ..,LIZ,PW=C,BR='CC',R='NSIO',SIO,COM0,..

   Clears the input buffer of parameter W bytes which are calculated in the COM1 directive, sets parameter W to parameter C, inserts a Backspace Record Command Chained into the IOCL queue and then inserts a Read command into the IOCL queue (the Read command is modified by the string argument NSIO which prevents the Start I/O from being executed), and then issues a Start I/O command, followed by a compare of the data input with a bias to the data in the output buffer that has the same bias.

7. ..,PO+4,PW=L,LBP,PC=W,J1/O<60,..

   Adds four to parameter O for the next bias to be used, sets parameter W to the byte count in parameter L, sets the output buffer to a binary progression, then sets the variable byte count in parameter C to parameter W, and jumps to directive statement one until parameter O is greater than 60.

8. ..,PO=4,PC-1,PC=W,J1/C=>8,..

   Resets the bias value in parameter O to four, decrements the variable byte count in parameter C by one, then sets the variable byte count in parameter C to parameter W and jumps to directive statement one until variable byte parameter C is less than eight.

9. ..,PG=0,PL=0,BAC,SM='RSGM',RWW,:

   Sets parameters G and L to zero, then resets the starting address to the input/output buffers to their original address by using the ..,BAC,.. directive, resets the Scatter/Gather mode register bit 7 by performing a Set Mode Register command modified by the string argument RSGM, and concludes by rewinding the device.

Example

The following shows the data values of the input/output buffers before and after the Scatter/Gather Read is performed.

- Consider that a bias of eight is being used
- Output buffer written to the drive has a binary progression

  000102030405060708090A0B0C0D0E0F10111213....
- Output buffer data pattern used to compare with input buffer

  000102030000000000405060700000000008090A0B

3.48 Test 47--Scatter/Gather Read Compare

This test verifies that the Scatter/Gather Read Compare capability is functioning properly. The test begins with a Scatter/Gather bias of four and increments this bias until a maximum bias of 56 is obtained. The data in the output buffer is manipulated to reflect the expect Read Compare data used by the command.

The test sequence follows:

1. RWW,PW=V,PW/2,PL=W,BCO,LBP,..

   Rewinds the device mechanism, sets parameter W to parameter V then divides by two, sets parameter L to parameter W, changes the starting address of the output buffer by using the ..,BCO,.. directive (which modifies the starting address of the output buffer by adding the data value of parameter L to the original buffer starting address), and then loads the top half of the output buffer with parameter W bytes of a binary progression.

2. ..,PL=0,BCO,LBP,PL=W,PG=0,BAC,..

   Sets parameter L to zero, then changes the starting address of the output buffer by using the ..,BCO,.. directive to the original starting address and loads the first (bottom) half of the output buffer with parameter W bytes of a binary progression, sets parameter L to parameter W, sets parameter G to zero, and restores the starting addresses of both the input and output buffers to their original values by using the ..,BAC,.. directive.

3. ..,PO=4,PC=32,1,PW=C,SM='RSGM',..

   Sets parameter O to four, sets parameter C to 32 (which is the decrementing byte count), sets parameter W to parameter C, sets the mode register modified by the string argument which resets the Scatter/Gather mode register bit 7 into the IOCL queue, and executes the Set Mode Register command.

4. ..,WRI,LIZ,SM='SGM,SGBC',..

   Performs the Write command, clears the input buffer to parameter W bytes, sets the mode register to the Scatter/Gather mode of operation (bit 7) and the bias in parameter O into the appropriate two bytes in the mode register data.

5. ..,LSC,BR='CC',RRC1,..

   Sets the output buffer to the correct data pattern (which is expected because of the variable bias in the mode register) by using the ..,LSC,.. directive, inserts a Backspace Record Command Chained into the IOCL queue, then inserts a Read Compare into the IOCL queue, and causes a Start I/O instruction to be executed.

6. ..,PO+4,PW=L,LBP,PW=C,..

   Adds four to the variable bias contained in parameter O, then sets the byte count in parameter W to the data in parameter L, sets the output buffer to the original binary progression, and then sets parameter W to the variable byte count in parameter C.

7. ..J1/O<=60,PO=4,PC-1,PW=C,J1/C=>8,..

   Jumps to directive statement one until parameter O (the variable bias) is greater than 60; otherwise, parameter C (the variable byte parameter) is decremented by one and parameter W is set of parameter C. This sequence continues until the variable byte count in parameter C is less than eight.

8. ..,PG=0,PL=0,BAC,SM='RSGM',RWW,:

   Sets parameters G and L to zero, then resets the starting address of the input/output buffers to their original address by using the BAC directive, resets the Scatter/Gather Mode Register bit 7 by performing a Set Mode Register command modified by the string argument RSGM and concludes by rewinding the device.

3.49 Test 48—Sync Chase Initial

This test performs an initial check of the Sync Chase capability. A binary progression of 256 bytes followed by a binary regression of 256 bytes is output to the drive. The test then sets the Sync Chase data word to FCFDFEFF and expects only the data from the binary regression to be input after the Read command is executed.

The test sequence follows:

1. RWW,PH=1,PS='FFFFFFFF',PU=4,PW=256,PW*2,PB=W,..

   Rewinds the device and sets a flag in parameter H, sets the mask to be used in the Sync Chase to parameter S and sets a counter for the number of times the Sync Chase test is to be performed into parameter U, sets the initial byte count of 256 to parameter W, then multiplies parameter W by two, and saves the data in parameter B.

2. ..,1,PU-1,PW=B,LBR,PW-256,LBP,PW*2,..

   Decrements the test counter in parameter U by one, sets parameter W to the byte count contained in parameter B, then sets the output buffer to a binary regression, subtracts 256 from the byte count, writes 256 bytes of a binary progression into the output buffer (effectively the output buffer has 256 bytes of a binary progression followed by 256 bytes of a binary regression), and multiplies the byte count in parameter W by two.

3. ..,CMW,PW=B,WR1,..

   Uses directive CMW, which shifts the mask by eight bytes after the initial mask of FFFFFFFF is used in the Sync Chase (subsequent masks are FFFFFF00, FFFF00FF, FF00FFFF), then sets the byte count in parameter W to the byte count stored in parameter B, and performs a Write command.

4. ..,CMW,SM='SW(FCFDFEFF),MWC,SWB(4)',..

   Sets the mode register with a Sync Chase pattern of FCFDFEFF with a byte count of four and sets the mask word in parameter C into the appropriate bytes of the Mode Register Sync Chase fields.

5. ..,PW=256,PW*2,LBR,LIZ,BRI,..

   Sets the byte count of 256 to parameter W, multiplies parameter W by two and conditions the output buffer to have a binary regression as the data in the output buffer which is compared to the data input if the sync chase pattern of FCFDFEFF is found, clears the input buffer, and performs a Backspace Record command.

6. ..,PW/2,R='SYC,SLI',J1/U>1,:

   Divides the byte count in parameter W by two, which causes only the binary regression to be input if the Read command is issued, then inserts a Read command into the IOCL queue which has the Sync Chase command flag ON and the Suppress Incorrect Length bit set, then issues a Start I/O instruction. The data is compared to determine that only the data after the FCFDFEFF is input from the record which is written with a binary progression followed by a binary regression. The program then jumps to direct state one, until the mask moves three times.

3.50 Test 49--Data Chain Sync Chase

This test verifies the Read Data Chain Sync Chase capability of the High-Speed Tape Processor. The diagnostic verifies this capability by outputting one record which has a binary progression, binary regression, binary progression, and binary regression to the drive. The Sync Chase word is set to the last four characters of the binary progression. After inputting data, the program verifies that only the data from the binary regression is input after the two Read Data Chain commands are executed.

The test sequence follows:

1. RWW,PW=256,PW*4,LBR,PW-256,LBP,PW-256,LBR,PW-256,LBP,..

   Rewinds the device, then sets parameter W to 256, multiplies parameter W by four and inserts a binary regression of parameter W bytes, subtracts 256 and inserts a binary progression into the output buffer, subtracts 256 and inserts a binary regression into the output buffer, and subtracts 256 and inserts a binary progression into the output buffer.

2. ..,PW*4,WRI,BRI,..

   Multiplies the byte count in parameter W by four, which enables the 256 bytes each of binary progression and binary regression to be written to the device, then inserts a Write command into the IOCL queue, issues a Start I/O instruction, then inserts a Backspace Record command into the IOCL queue which positions the device for the Read command.

3. ..,SM='SW(FCFDFEFF),MW(FFFFFFFF),SWB(4)',..

   Sets the mode register with a Sync Chase pattern of FCFDFEFF with a byte count of four and sets the mask word to FFFFFFFF into the appropriate bytes of the Mode Register Sync Chase field.

4. ..,PW=256,PW*2,LBR,LIZ,..

   Sets parameter W to 256 and multiplies the parameter by two, which is used to condition the output buffer with two patterns of a binary regression, sets the output buffer to the two binary regressions, and then clears the input buffer.

5. ..,PW/2,R='SYC,DC,SLI',R='SYC,SLI',:

   Divides parameter W by two to obtain the correct number of bytes for the two Read Data Chained commands, then inserts a Read command which is data chained and has the Suppress Incorrect Length and Sync Chase bits set in the command flags into the IOCL queue, then inserts a Read command into the IOCL queue which has the command flags of Suppress Incorrect Length and Sync Chase set which causes a Start I/O instruction to be executed. The input data is checked for two binary regressions as the correct data.

3.51 Test 50—Variable Sync Chase

This test verifies the capability of High-Speed Tape Processor to correctly perform a Read Sync Chase command on a variable sync word in a binary progression.

The test sequence follows:

1. RWW,PW=256,PF=W,PF-4,PW*2,PB=W,..

   Rewinds the device, then sets parameter W to 256 and sets parameter W to parameter F (this is the moving pointer which changes if data is compared on read operations), subtracts four from parameter F, then multiplies parameter W by two, and sets parameter B to the data in parameter W.

2. ..,1,PU=4,PH=1,PS='FFFFFFFF',..

   Sets parameter U to four (parameter U determines how many times the mask is shifted), sets parameter H to one (parameter H is used as a flag in the ..,CMW,.. directive), and sets parameter S to an initial mask of FFFFFFFF.

3. ..,2,PL=0,BCO,PW=B,LBR,PW-256,LBP,..

Parameter L is set to zero and the starting address of the output buffer is reset to the original starting address of the diagnostic, parameter W is set to the data in parameter B, which is equal to 512, sets the output buffer to a binary regression of W bytes, then subtracts 256 from parameter W, and conditions the first 256 bytes of the output buffer with a binary progression. Effectively, the output buffer has 256 bytes of binary progression, followed by 256 bytes of a binary regression.

4. ..,PW=B,WRI,CMW,..

Sets parameter W with the contents of parameter B, which is equal to 512, inserts a Write command into the IOCL queue, and issues a Start I/O instruction, then uses the CMW directive, which determines the type of mask that is used by the Sync Chase Read.

5. ..,PU-1,SM='SMF,MWC,SWB(4)',..

Subtracts one from the shift mask counter in parameter U, sets the mode register to a Sync Chase byte count of four and a mask word controlled by parameter C and a sync word controlled by parameter F.

6. ..,PW=B,LIZ,BRI,PW-256,PL=F,PL+4,BCO,..

Sets parameter W to the original byte count in parameter B and clears the input buffer, then backspaces a record and decrements the byte count in parameter W by 256, sets parameter L to parameter F, which determines what part of the output buffer is used for the data compare. Then adds four to position the output buffer address properly, and performs a ..,BCO,.. directive, which establishes the new output buffer address.

7. ..,R='SYC,SLI',J2/U>1,PF-1,PU=4,J1/F>=8,PG=0,BAC,RWW,:

Inserts a Read command with the Sync Chase and Suppress Illegal Length command flags into the IOCL queue and executes a Start I/O instruction. The data input is verified by comparing the input data to the data which is being pointed to in the output buffer. The program jumps to statement two if parameter U is greater than one; otherwise, parameter F is decremented by one thereby causing the next four bytes to be used as a sync word, parameter U is reset to four to allow testing the four masked words used by the diagnostic and the program jumps to directive statement one if parameter F, which is the sync word pointer, is greater than eight. Otherwise, the program sets parameter G to 0, resets the original starting address of the input/output buffers, rewinds the device and exits the test.

Example

The following is an example of the input/output data buffers used in this test.

- Output buffer before Write command is issued

0001020304050607..F8FCFDFEFFFFFEFDFCFBFA..... 0403020100

- Sync Chase Word is FCFDFEFF
- The starting address of the output buffer is changed to the location which points to FFFFFEFD
- After the Read command is performed, the input/output data compare begins with the starting address of the input buffer and the new calculated address of the output buffer to determine that only the data after the sync word is inputted if the Read command is executed.

CHAPTER 4

ERROR REPORTING

4.1 Error Message Assignment

The Gould RPU High-Speed Tape Processor Diagnostic writes all error messages to the listed output (LO) file. The default LO file assignment is to the operator's console.

To assign the LO file to a device other than the operator's console, enter:

```
DXP> ASSIGN LO=ttaa[,dsa]
```

ttaa — specifies the four-character device mnemonic containing a two-character device type code and a two-character hexadecimal channel address.

dsa — specifies an optional two-character hexadecimal device subaddress.

4.2 Error Messages and Message Sequence

All error message sequences consist of:

1. The message separation banner

```
<<<<'taskname'----hh:mm:ss>>>>
```

taskname — specifies the one to eight character name indicated in the load map.

hh:mm:ss — specifies the system elapsed execution time or the current time-of-day (if initialized) at the point of error.

2. The error message header

```
TST:ttt,CA:cc,DSA:ss,PC:pppp
```

ttt — specifies the decimal test number of the erring test.

cc — specifies the hexadecimal High-Speed Tape Processor address assigned.

ss — specifies the hexadecimal device subaddress assigned.

pppp — specifies the program counter value within the erring test.

3. The status error message

```
STATUS ERROR

EXP (UNIT)s (IOCD) bbbbbb (BITS) eeee (COUNT) tttt
ACT (UNIT)s (IOCD) bbbbbb (BITS) eeee (COUNT) tttt
```

EXP	indicates the expected status values.
ACT	indicates the actual status values.
s	specifies the device subaddress.
bbbbbb	specifies the expected and actual data addresses for the IOCL termination.
eeee	specifies the status of the device.
tttt	specifies the hexadecimal remaining byte count from the last IOCD executed in the IOCL queue.

4.3 Status Error Report Exceptions

Status word checking is performed after all Start I/O (SIO) instructions are completed. If any discrepancies are detected between the expected status and the actual status obtained from the device, the status error(s) is reported. However, status words are not checked if any of the following error conditions are detected after issuing a Start I/O instruction:

- Condition Code Errors
  If condition code errors occur, they are reported with no further status checking.

- No Interrupt or Device End Receive Error
  If an expected interrupt is not detected from the device, the no status interrupt message is reported with no further status checking.

- Status Address Word Errors
  If the status address is not correct, the program does not check the location for valid status information.

4.4 Condition Code Errors

Condition code errors are reported if they are detected during execution of an extended I/O instruction. Normal condition codes of request, accepted and queued, hexadecimal 8, or status stored hexadecimal 4 are not reported. The following messages are reported if an error is detected:

If a condition code of hexadecimal 0 is detected and the current instruction being executed is not a Reset Channel instruction, the following message is written:

```
CC=0, UNASSIGNED
```

This condition code is valid only for the Reset Channel instruction.

If the channel times out and remains busy without allowing for a request accepted or status stored condition, the following message is written:

CC=1, CHANNEL BUSY

If the channel/controller is not defined, the following message is written:

CC=2, UNDEFINED

If the subchannel times out as busy and does not allow for a request accepted or status stored condition, the following message is written:

CC=3, SUBCHANNEL BUSY

If the controller is rejecting the current command as not being supported by the device, the following message is written:

CC=5, UNSUPPORTED

If an unassigned condition code is detected, the following message is written:

CC=x, UNASSIGNED

x = hexadecimal 6,7,9,10,11,12,13,14,15

These condition codes are unassigned and should never be received.

If any of the above errors are reported, no attempt is made to check the status information, because there should be no status information received from the controller.

4.5 No Interrupt or Device End Received Error Message

If an interrupt is expected and not received from the device or no device end is reported after an interrupt is received from the device, one of the following messages is reported:

If no interrupt is received from the device but an interrupt is expected, the following message is written:

CC=c, NO SI RECEIVED

c specifies the condition code.

Usually, this error message follows a condition code error message indicating that the instruction is never executed because the IOM/controller is not able to accept and/or queue the particular instruction.

If an interrupt is received from the device, but no device end is received as a status condition, the following message is written:

CC=c, NO DEVICE END

c specifies the condition code.

If any of the above messages is reported, no status information or sense data information is checked for validity.

4.6 Status Address Word Error

Status addresses obtained in the service interrupt receiver area are checked for their validity with respect to the data address used in the Initialize Channel (INCH) command. Only two valid addresses are accepted with all others being reported as errors. The two valid addresses are as follows:

- INCH Command Data address
- INCH Command Data address +8 bytes

Any other addresses cause the following message to be written:

```
EXP STATUS ADDRESS cfffffff OR csssssss
ACT STATUS ADDRESS caaaaaaa
```

c	specifies the condition code setting in the status word.
fffffff	specifies the first valid data address as presented by the INCH command.
sssssss	specifies the second valid data address as the INCH data address plus eight bytes.
aaaaaaa	specifies the actual address received in the service interrupt receiver.

If this error message is reported, no verification is made as to the validity of the device status contents. Therefore, no comparisons are made between the expected and actual status received from the device.

4.7 Program Start Errors

After receiving the first Start command, the program performs a Reset Channel, Enable Channel Interrupt, and a Start I/O instruction to initialize the IOM/controller. The following error messages are related to the first Start command:

If a Reset Channel instruction failure occurs, the following message is written:

```
TST:RCH,CA:cc,DSA=ss,PC:ppppp
aaaaa,DRTV:RCH
CMND:iiiiiiii
```

cc	specifies the hexadecimal High-Speed Tape Processor address assigned.
ss	specifies the hexadecimal device subaddress assigned.
ppppp	specifies the hexadecimal program counter value.
aaaaa	specifies the address of the command device instruction.
iiiiiiii	specifies the command device instruction issued.

This message usually is accompanied by an incorrect condition code error message.

If an Enable Channel Interrupt (ECI) instruction failure occurs, the following message is written:

```
TSI:ECI,CA:cc,DSA:ss,PC:ppppp
aaaaa,DRTV:ECI
CMND:iiiiiiii
```

This message usually is accompanied by an incorrect condition code error message.

If an Initialize Channel (INCH) instruction failure occurs, the following message is written:

```
TST:ICH,CA:cc,DSA:ss,PC:ppppp
aaaaa,DRTV:ICH
CMND:iiiiiiii IOCD dddddddd dddddddd
```

dddddddd specifies the two-word Initialize Channel IOCD.

This message usually is accompanied by an incorrect condition code error message, a no status interrupt received message, and the following message:

```
TERMINATED:  CANNOT INITIALIZE CHANNEL
```

This message indicates that the device cannot be initialized and the program does not proceed any further until the device is initialized.

If a Set Mode Register command failure occurs, the following message is written:

```
TST:SET,CA:ss,PC:ppppp
aaaaa,DRTV:SET
CMND:iiiiiiii IOCD dddddddd dddddddd
```

This message usually is accompanied with sense data information indicative of the type of error which occurs.

4.8 Data Error Reporting

The data error message follows:

```
DATA ERROR(S)
FR#:BFO:BFI:
ffff dd   dd
 .    .    .
 .    .    .
 .    .    .
ffff dd   dd
xxxx OUT OF yyyy
```

FR# indicates the frame number(s) in error.

BFO indicates the output buffer (expected) value(s).

BFI indicates the input buffer (actual) value(s).

ffff specifies the decimal frame number of the first frame in error and each successive error up to five frames in error.

dd specifies the hexadecimal data value contained in the indicated buffer segment.

xxxx specifies the decimal total number of frames in error.

yyyy specifies the decimal total number of frames compared.

The data values listed under the BFO column identify the data written to the device. However in Tests 45 through 50, these data values represent the data expected from the read operation in the Scatter/Gather and Sync Chase modes of operation.

One of the following messages is written for each occurrence of a status bit in either the expected or actual status unless option 12 (Inhibits error message expansion) is set. If the message is preceded by a plus (+) sign, the status bit indicated is received but not expected, and if preceded by a minus (-) sign, the status bit indicated is expected but not received.

STATUS BIT

00	HIO AND STOP FUNCTION
01	PROGRAM CONTROLLED INTERRUPT
02	INCORRECT LENGTH
03	CHANNEL PROGRAM CHECK
04	CHANNEL DATA CHECK
05	CHANNEL CONTROL CHECK
06	INTERFACE CHECK
07	CHAINING CHECK
08	DEVICE BUSY
09	STATUS MODIFIER
10	NOT USED
11	NOT USED
12	CHANNEL END
13	DEVICE END
14	UNIT CHECK
15	UNIT EXCEPTION

One of the following messages is written for each occurrence of a sense data bit in either the expected or actual status unless option 12 (Inhibits error message expansion) or option 14 (Inhibits Sense error messages) is set. If the message is preceded by a plus (+) sign, the sense data bit indicated is received but not expected, and if preceded by a minus (-) sign, the sense data bit indicated is expected but not received.

SENSE WORD ZERO

SENSE BIT

00	COMMAND REJECT
01	INTERVENTION REQUIRED
02	BIT 2 NOT USED
03	EQUIPMENT CHECK
04	DATA CHECK
05	DATA OVER-UNDER RUN

SENSE BIT

06	BIT 6 NOT USED
07	BIT 7 NOT USED
08	FORMATTER REJECT **
09	OPERATION INCOMPLETE
10	END-OF-FILE
11	CORRECTABLE ERROR
12	HARD ERROR
13	BIT 13 NOT USED
14	BIT 14 NOT USED
15	BIT 15 NOT USED
16	MODE BIT NOT USED 0
17	MODE BIT NOT USED 1
18	MODE BIT NOT USED 2
19	SCATTER/GATHER MODE
20	MODE BIT NOT USED 4
21	MODE BIT NOT USED 5
22	MODE BIT NOT USED 6
23	MODE REGISTER BIT 7
24	DRIVE READY
25	DRIVE ON-LINE
26	DRIVE IS FILE PROTECTED
27	DRIVE IN NRZI
28	BIT 28 NOT USED
29	DRIVE IS AT LOAD POINT
30	DRIVE IS AT END-OF-TAPE
31	GCR MODE, SENSE BIT 31

Sense words one and two are not written to the listed output (LO) file if option 18 (Inhibits writing error bit analysis messages) is set.

SENSE WORD ONE

SENSE BIT

00	DTP
01	CRC ERROR
02	TACH
03	WRITE STATUS
04	CRC-F (P)
05	SNS WD1, BIT 5 NOT USED
06	SNS WD1, BIT 6 NOT USED
07	SNS WD1, BIT 7 NOT USED
08	DEAD TRACK 0
09	DEAD TRACK 1
10	DEAD TRACK 2
11	DEAD TRACK 3
12	DEAD TRACK 4
13	DEAD TRACK 5
14	DEAD TRACK 6
15	DEAD TRACK 7
16	DIAG MODE
17	VELOCITY ERROR
18	END DATA CHECK

SENSE BIT

19	SPARE WD1 BIT 19
20	MTE
21	PARTIAL RECORD
22	UCE
23	WTM CHECK
24	FORMATTER REJECT CODE BIT 0
25	FORMATTER REJECT CODE BIT 1
26	FORMATTER REJECT CODE BIT 2
27	FORMATTER REJECT CODE BIT 3
28	FORMATTER REJECT CODE BIT 4
29	FORMATTER REJECT CODE BIT 5
30	FORMATTER REJECT CODE BIT 6
31	FORMATTER REJECT CODE BIT 7

Combinations of sense bits 24 through 31 of sense word one define formatter reject codes. Refer to the section, Formatter Reject Code Messages, for messages associated with the various codes.

SENSE WORD TWO

SENSE BIT

00	ON LINE STATUS
01	READY STATUS
02	HIGH DENSITY
03	BACKWARD STATUS
04	FILE PROTECT
05	WRITE INHIBIT
06	BOT STATUS
07	EOT STATUS
08	CRC-F 0
09	CRC-F 1
10	CRC-F 2
11	CRC-F 3
12	CRC-F 4
13	CRC-F 5
14	CRC-F 6
15	CRC-F 7

4.9 Additional Error Messages

If no interrupt is generated and the device is inactive during a Halt I/O instruction, the following message is written:

IDLE STATE: NO INTERRUPT AFTER HIO

This message is exclusive to Test 2.

If no interrupt is generated and the device is inactive during a Stop I/O instruction, the following message is written:

```
IDLE STATE: NO INTERRUPT AFTER STP
```

This message is exclusive to Test 2.

If a discrepancy occurs in the expected and actual condition codes detected by the specific test, the following message is written:

```
ACTIVE STATE: EXPECTED CC e
ACTIVE STATE: ACTUAL CC a
```

e specifies the expected condition code for the test being executed.

a specifies the actual condition code for the test being executed.

This message is exclusive to Test 2 and Test 44 if a Test I/O instruction is being executed.

If the program cannot perform an Initialize Channel command to the device, the following message is written:

```
CANNOT INITIALIZE CHANNEL
```

This message is accompanied by a condition code error message or a no status interrupt message. No further testing is performed until the channel is initialized.

If no interrupt is detected but an interrupt is expected during the execution of interrupt instruction testing, the following message is written:

```
NO SI
```

This message is exclusive to Test 1.

If an interrupt is detected but an interrupt is not expected during the execution of interrupt instruction testing, the following message is written:

```
UNEXPECTED SI
```

This message is exclusive to Test 1.

If a reset channel directive is performed but no initialize channel command is issued before attempting to perform another I/O directive, the following message is written:

```
CHANNEL MUST BE INITIALIZED
```

This message is exclusive to Test 11.

If during the execution of Test 11, a Write Channel instruction is issued to the device and the unsupported transaction condition code (CC=5) is not detected after the instruction is executed, the following message is written:

```
WCWCS-NO UNSUPPORTED TRANS STATUS
```

If during the execution of Test 11B, a Grab Controller instruction is issued to the device and the unsupported transaction condition code (CC=5) is not detected after the instruction is executed, the following message is written:

GRIO=NO UNSUPPORTED TRANS STATUS

This message is exclusive to Test 11.

If during the execution of Test 11, a Reset Controller instruction is issued to the device and the unsupported transaction condition (CC=5) is not detected after the instruction is executed, the following message is written:

RSCTL-NO UNSUPPORTED TRANS STATUS

If a specified command is executed and an end-of-media (EOM) sense bit is received, the following message is written:

CC=x END OF MEDIA

x specifies the condition code received from the device.

4.10 Formatter Reject Code Messages

One of the following messages is written depending on the value of sense bits 24 through 31 of the word one. The bit pattern defines which of these messages is written.

FORMATTER REJECT CODE	MESSAGE
10 (bit 27 set)	THE ADDRESSED MAGNETIC TAPE UNIT DID NOT GO TO READ STATUS
20 (bit 26 set)	THE FORMATTER HAS DETECTED AN UNIMPLEMENTED WORD IN ITS INTERNAL MICROPROGRAM
28 (bits 26 and 28 set)	THE AUTOMATIC READ AMPLIFICATION (ARA) BURST PORTION OF THE GROUP CODED RECORDING (GCR) INDENTIFICATION BURST JUST WRITTEN DID NOT HAVE ALL NINE TRACKS ACTIVE
30 (bits 26 and 27 set)	THE ADDRESSED MAGNETIC TAPE UNIT DID NOT GO TO WRITE STATUS AFTER A WRITE TYPE COMMAND WAS INITIATED
38 (bits 26, 27, and 28 set)	THERE WAS NO INTER-RECORD GAP (IRG) DETECTED FOLLOWING THE AUTOMATIC READ AMPLIFICATION BURST
40 (bit 25 set)	THE FORMATTER HAS DETECTED WRONG PARITY FOR ONE OF ITS INTERNAL MICROPROGRAM WORDS

FORMATTER REJECT CODE	MESSAGE
48 (bits 25 and 28 set)	WHEN THE (PE) OR (GCR) IDENTIFICATION BURST JUST WRITTEN WAS RECHECKED ON A READ, IT WAS ON THE WRONG TRACK
50 (bits 25 and 27 set)	THE ADDRESSED MAGNETIC TAPE UNIT DID NOT DROP WRITE INHIBITED STATUS
58 (bits 25, 27, and 28 set)	THE ADDRESSED MAGNETIC TAPE UNIT FAILED TO INITIATE TAPE MOTION
60 (bits 25 and 26 set)	THE ADDRESSED MAGNETIC TAPE UNIT DID NOT GO ONLY TO ERASE STATUS
70 (bits 25, 26, and 27 set)	THE ADDRESSED MAGNETIC TAPE UNIT DID NOT GO TO BACKWARD STATUS
78 (bits 25, 26, 27, and 28 set)	THE AUTOMATIC READ AMPLIFICATION (ARA) IDENTIFICATION WAS UNREADABLE DURING A (GCR) READ OR WRITE COMMAND
80 (bit 24 set)	THE ADDRESSED MAGNETIC TAPE UNIT IS NOT IN READY STATUS
88 (bits 24 and 28 set)	THE ADDRESSED MAGNETIC TAPE UNIT RESET READY STATUS
90 (bits 24 and 27 set)	THE MAGNETIC TAPE SYSTEM (MTS) DOES NOT HAVE (NRZI) CAPABILITY, BUT IT ATTEMPTED TO READ AN (NRZI) TAPE OR THE (MTS) DOES NOT HAVE (NRZI) CAPABILITY AND WAS UNABLE TO READ A (PE) OR (GCR) IDENTIFICATION BURST DURING EITHER A READ OPERATION OR A READ CHECK AFTER WRITING THE IDENTIFICATION BURST OR THE (MTS) HAS (NRZI) CAPABILITY, BUT WAS UNABLE TO READ A (PE) OR (GCR) IDENTIFICATION BURST DURING A CHECK AFTER WRITING THE IDENTIFICATION BURST
A0 (bits 24 and 26 set)	THE ADDRESSED MAGNETIC TAPE UNIT WAS IN FILE PROTECT STATUS WHEN A WRITE COMMAND WAS ATTEMPTED

FORMATTER REJECT CODE	MESSAGE
A8 (bits 24, 26, and 28 set)	AN (IRG) LONGER THAN 25 FT IN (PE) OR (NRZI) MODE OR LONGER THAN 15 FT IN (GCR) MODE WAS DETECTED ON A READ OR SPACE TYPE COMMAND
B0 (bits 24, 26, and 28 set)	DURING A BACKWARD MOTION AFTER WRITING THE IDENTIFICATION BURST, THE BEGINNING-OF-TAPE (BOT) WAS NOT REACHED IN THE DISTANCE EXPECTED
B8 (bits 24, 26, 27, and 28 set)	THE MAGNETIC TAPE ATTEMPTED TO BACKSPACE OVER A BAD RECORD JUST WRITTEN, BUT WAS UNABLE TO DETECT THE RECORD
C0 (bits 24 and 25 set)	THE TRANSFER ACKNOWLEDGE (TRAK) RESPONDS TO INITIATING TRANSFER REQUESTS (TREQS) WERE NOT RECEIVED WITHIN 75 MSEC ON A WRITE TYPE COMMAND
C8 (bits 24, 25, and 28 set)	A BACKWARD TYPE COMMAND (EXCEPT A REWIND OR A REWIND UNLOAD) WAS GIVEN, BUT TAPE WAS ALREADY POSITIONED AT (BOT)
D0 (bits 24, 25, and 27 set)	THE ADDRESSED MAGNETITC TAPE UNIT IS NOT IN ON-LINE STATUS
D8 (bits 24, 25, 27, and 28 set)	DURING THE READ BACK CHECK OF A WRITE OPERATION DATA WAS DETECTED IN THE (IRG) AREA
E8 (bits 24, 25, 26, and 28 set)	THE ADDRESSED MAGNETIC TAPE UNIT FAILED TO RESET STATUS DURING A REWIND OPERATION
F0 (bits 24, 25, 26, and 27 set)	NOISE (POSSIBLY DATA) WAS DETECTED DURING EITHER AN ERASE GAP COMMAND OR A WRITE COMMAND FOLLOWING A READ TYPE COMMAND
F8 (bits 24, 25, 26, 27, and 28 set)	DURING THE READ BACK CHECK OR A WRITE OR WRITE TAPE MARK COMMAND, NO DATA WAS DETECTED
IMSG	INVALID FORMATTER REJECT CODE = xx**

CHAPTER 5

DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Compiler

The Gould RPU High-Speed Tape Processor Diagnostic (RP.HTP.r) is structured around the capabilities of the Directive Language Compiler. The Directive Language Compiler (a component of the Diagnostic Executive Program), and its associated run-time package (a component of the diagnostic) use directive language to offer the following diagnostic capabilities:

- The ability to reduce a diagnostic test into its failing components for troubleshooting.
- The ability to create new directive test sequences.

The basic component of the directive language is the directive. For all Level Two diagnostics, there are two categories of directives -- standard and unique. The DEXP provides standard directives for all Level Two diagnostics, while each Level Two diagnostic contains unique directives to perform specific functions within the individual program.

A complete understanding of standard and unique directives is needed to use the directive language effectively.

5.2 DEXP Standard Directives

The Gould Diagnostic Executive Program contains nine standard directive categories. The following is a list of standard directives and their functions.

5.2.1 "A" Directives (Assign)

Three "A" directives assign channel, controller, and unit addresses for each controller.

..,A,..

Uses the channel address and the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,AC,..
..,ACn,..
..,ACA,..
..,ACAn,..

Uses the specified channel address n or, if n is not specified, the channel address currently in the device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,ADSA,..
..,ADSAn,..
..,AU,..
..,AUn,..

Uses the specified device subaddress n, or if n is not specified, the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

5.2.2 "C" Directives (Cycle)

Four "C" directives cycle the output buffer (equivalent to a circular shift with end-to-end wraparound).

..,CL,..
..,CLn,..

Cycles the output buffer left by number n byte positions. If number n is not specified, the output buffer is cycled left by one byte position.

..,CR,..
..,CRn,..

Cycles the output buffer right by number n byte positions. If number n is not specified, the output buffer is cycled right by one byte position.

..,CLp,..

Cycles the output buffer left by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled left by one byte position.

..,CRp,..

Cycles the output buffer right by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled right by one byte position.

5.2.3 "D" Directives (Dump)

Four "D" directives dump all or any specified part of the diagnostic's input buffer, output buffer, or parameter set to the listed output (LO) file. If the LO file is assigned to a line printer, the byte count is rounded up to the next multiple of 32 bytes (eight words). Otherwise, the byte count is rounded up to the next multiple of 16 bytes (four words).

..,DI,..
..,DIn,..

Dumps number n or, if number n is not specified, parameter W bytes of the diagnostic's input buffer. If number n is not specified and parameter W equals zero, the entire input buffer is dumped.

..,DO,..
..,DOn,..

Dumps number n or if number n is not specified, parameter W bytes of the diagnostic's output buffer. If number n is not specified and parameter W equals zero, the entire output buffer is dumped.

..,DP,..

Dumps the contents of all parameters (A through Z).

..,DPp,..

Dumps only the contents of parameter p (any specified parameter A through Z).

5.2.4 "H" Directives (Hold)

Two "H" directives hold diagnostic execution. These directives establish delays between directives.

..,H,..
..,Hn,..

Holds diagnostic execution for the specified number of time units n or, if number n is not specified, for the number of time units in parameter X. If parameter X contains zero, the hold duration is one time unit (determined by the Real-Time Clock rate). If the Real-Time Clock rate is 120 Hz, one time unit equals 8.3 ms.

..,HS,..
..,HSn,..

Holds diagnostic execution for the specified number of seconds n or, if number n is not specified, for the number of seconds in parameter X. If parameter X contains zero, the hold duration is 1 second.

5.2.5 "J" Directives (Jump)

Five "J" directives transfer control in a directive string. The following are the relational operators to use with conditional jumps:

(op):		
	-	not
	=	equal to
	>	greater than
	<	less than
	=>	equal to or greater than
	>=	greater than or equal to
	<=	less than or equal to
	=<	equal to or less than

..,Jn,..

Jumps indefinitely to statement number n (0<=n<=9).

..,JZ,..

Jumps indefinitely to the statement number in parameter Z.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k,..

Jumps to statement number n or the statement number in parameter Z, k times (1<k<=32,767).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p,..

Jumps to statement number n or the statement number in parameter Z the number of times defined by the contents of the specified parameter (A through Z).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k,..

Jumps to the statement number n or the statement number in parameter Z as long as the contents of p (any specified parameter A through Y) satisfies the stated relationship (op) with the constant k.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2,..

Jumps to statement number n or the statement number in parameter Z as long as the contents of p_1 (the first specified parameter A through Z) satisfies the stated relationship (op) with the contents of p_2 (the second specified parameter A through Y).

5.2.6 "L" Directives (Load)

Fifteen "L" directives load input or output buffers with a variety of data test patterns.

..,L=' ',..

Loads the output buffer with 'any literal value' and stores the literal string length in parameter W.

..,LA,..
..,LAn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the 64-character printable ASCII subset. If parameter W equals zero, the entire output buffer is loaded.

..,LIZ,..
..,LIZn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with all zeros. If parameter W equals zero, the entire input buffer is loaded.

..,LZ,..
..,LZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all zeros. If parameter W equals zero, the entire output buffer is loaded.

..,LO,..
..,LOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all ones. If parameter W equals zero, the entire output buffer is loaded.

..,LFZ,..
..,LFZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating zero (FE,FD,FB,...,BF,7F,FE,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LFO,..
..,LFOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating one (01,02,04,...,40,80,01,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LBK,..
..,LBKn,..

Loads number n or, if number n is not specified, parameter W bytes to the output buffer with ASCII blanks (20,20,20,...,20,20,20,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBP,..
..,LBPn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary progression (00,01,02,...,FE,FF,00,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBR,..
..,LBRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary regression (FF,FE,FD,...,01,00,FF,...). If parameter W equals zero, the entire output buffer is loaded.

..,LIR,..
..,LIRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the initial random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LLR,..
..,LLRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the last random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LNR,..
..,LNRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the next random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LOI,..
..,LOIn,..

Loads number n or if number n is not specified, parameter W bytes of the output buffer with the contents of the input buffer. If parameter W equals zero, the entire output buffer is loaded.

..,LIO,..
..,LIOn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with the contents of the input buffer. If parameter W equals zero, the entire output buffer is loaded.

5.2.7 "P" Directives (Parameter)

Five "P" directives manipulate entries in the diagnostic's parameter table. The following are arithmetic and logical operators to use with "P" directives:

(op):		
	=	replacement
	+	addition
	-	subtraction
	*	multiplication
	/	division
	.A.	logical AND
	.O.	logical OR
	.E.	logical EXCLUSIVE OR
	.N.	two's complement
	.C.	one's complement
	>	circular shift right
	<	circular shift left

..,Pp(op)k,..

The contents of the specified parameter (p specifies a parameter A through Z) is fetched then the logical OR arithmetic operation is performed. The result is stored into the specified parameter (the range for k is 0 through 32,767).

..,Pp_1(op)p_2,..

The contents of the specified parameters p_1 and p_2 are fetched, then the logical OR arithmetic operation is performed. The result then is stored in the specified parameter p_1.

..,PD(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of parameter D (and the second parameter if also specified) are fetched, then the logical OR arithmetic operation is performed. The result then is stored in parameter D and also into the entire output buffer.

..,PI(op) $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of parameter I and either the second parameter Pp (if specified) or k are fetched then the logical OR arithmetic operation is performed. The result is stored into parameter I and also into word one of the output buffer.

..,PZ= $\left\{\begin{matrix} k \\ Pp \end{matrix}\right\}$,..

The contents of Pp (P specifies any parameter A through Y) or k, is stored into parameter Z. The value of parameter Z represents a statement number to which the ..,JZ,.. directive transfers control. The only valid operator for this directive is = (replacement). The contents of the specified parameter must be between zero and nine.

5.2.8 "S" Directive (Stop)

The "S" directive creates convenient breaks in a directive string for performing necessary functions.

..,STPn,..

Stops diagnostic execution and issues the following prompt to the operator communications (OC) file:

```
'taskname' ,STPn, DIRECTIVE ; [C,H,T] ?
```

5.2.9 "T" Directive (Time)

The "T" directive writes the time-of-day and an informative message (if specified in the directive) to the listed output (LO) file.

..,TIME,..
..,TIME (message),..

5.3 Gould RPU High-Speed Tape Processor Unique Directives

The Gould RPU High-Speed Tape Processor Diagnostic contains unique directives to perform specific functions within the diagnostic. Many unique directives, however, cross diagnostic boundaries and are used in different diagnostics.

The following is a list of the Gould RPU High-Speed Tape Processor unique directives:

5.3.1 "A" Directives

..,AFn,..
..,AF='CC',..

Inserts an Advance File I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic advances the number of files specified by n, or if command chain 'CC' is specified, IOCD word 2, bit 1, is set, and the next directive is processed.

..,AR,..
..,ARn,..
..,AR='CC',..

Inserts an Advance Record (skip record) I/O command doubleword (IOCD) into the I/O command list (IOCL) queue. The diagnostic advances the number of files specified by n, or if n is not specified, the number specified by parameter Y.

If command chain 'CC' is specified, IOCD word 2, bit 1, is set, the record count defaults to one, and the next directive is processed.

5.3.2 "B" Directives

..,BAC,..

Manipulates the starting addresses of the input and output buffers. The data value in parameter G at the time this directive is executed is added to the starting address of the input and output buffers and is used by the diagnostic for any Write/Read Buffer Compare operations. The input/output buffer starting addresses are reset always to their original data addresses if the diagnostic is started or restarted.

..,BCI,..

Manipulates the starting address of the input buffer. The data value in parameter K at the time this directive is executed is added to the starting address of the input buffer. The input buffer starting address is reset always to the original data address if the diagnostic is started or restarted.

..,BCO,..

Manipulates the starting address of the output buffer. The data value in parameter L at the time this directive is executed is added to the starting address of the output buffer. The output buffer starting address is reset always to the original data address if the diagnostic is started or restarted.

..,BFn,..
..,BF='CC',..

Inserts a Backspace File I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic backspaces the number of files specified by n, or if command chain 'CC' is specified, IOCD word 2, bit 1, is set, and the next directive is processed.

..,BFI='address',..

Relocates the diagnostic's input buffer to the specified address.

..,BFI='PGM',..

Relocates the diagnostic's input buffer to the default location in the diagnostic.

..,BFO='address',..

Relocates the diagnostic's output buffer to the specified address.

..,BFO='PGM',..

Relocates the diagnostic's output buffer to the default location in the diagnostic.

..,BRn,..
..,BR='CC',..

Inserts a Backspace Record I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic backspaces the number of records specified by n, or if command chain 'CC' is specified, IOCD word 2, bit 1, is set, and the next directive is processed.

5.3.3 "C" Directives"

..,CMW,..

Initially sets parameter C to the value in parameter S and establishes this data value as the Sync Mask Word, which may be used in the Sync Chase mode of operation. The directive uses parameter H as a flag and initially sets the original data in parameter S to the Sync Chase Mask Word. Parameter S then has the least significant byte set to zero, and subsequent calls to this directive shift this zero byte into successively higher order bytes. This directive is used in the Sync Chase tests (Tests 48 through 50).

..,COM,..

Calculates the number of bytes needed to perform a data compare of the input buffer to the output buffer. This directive performs the calculation and then branches to the data compare subroutine in the diagnostic. This directive is used in Test 56 to perform a function needed for special testing of the input/output buffers.

..,COM1,..

Calculates the number of bytes needed to perform a data comparison of the input buffer to the output buffer. However, the calculated byte count is stored in parameter W and the data compare subroutine is not called.

5.3.4 "D" Directives

..,DCC,..

Dumps the last condition code received by the diagnostic as a result of an extended I/O instruction.

..,DQ,..

Dumps the contents of the I/O command list (IOCL) queue to the listed output (LO) file in the following format:

```
IOCD(ii):ooffcccc 00bbbbb
```

ii	specifies the IOCD queue slot number in the IOCL queue.
oo	specifies the contents of the order byte in IOCD word one.
ff	specifies the contents of the flag byte in IOCD word one.
cccc	specifies the hexadecimal value of the initial byte count in IOCD word one.
bbbbb	specifies the hexadecimal value of the initial buffer address in IOCD word two.

If no IOCL queue entry exists, the diagnostic writes the following message:

```
IOCL QUEUE EMPTY
```

..,DSD,..

Dumps the sense data received from the last executed Sense command to the LO file.

..,DSG,..

Dumps the contents of mode register bytes 1 through 11 to the LO file in the following format:

Message	Position in Mode Register
SCATTER/GATHER BIAS **	(MODE REGISTER BYTE 1-2)
SYNC WORD BYTE COUNT **	(MODE REGISTER BYTE 3)
SYNC WORD ********	(MODE REGISTER BYTE 4-7)
MASK WORD ********	(MODE REGISTER BYTE 8-11)

**	specifies the hexadecimal data in the bytes indicated.

..,DST,..

Dumps the channel/device status received from the last executed Start I/O (SIO) instruction to the LO file.

..,DUMP='arg_1, arg_2,...,arg_n',..

Dumps the geometry, status, and (optionally) the data of a magnetic tape to the LO file. The arguments for the directive specify the particular files/records to be dumped, or if the GEOM (geometry) argument is specified, the entire tape is dumped.

The geometry of a tape is the number of files on the tape, the number of records per file, and the size (in bytes) of each record.

Legal arguments for this directive are: RW, GEOM, RR(n), AR(n), BR(n), AF(n), and BF(n).

The status returned by the directive may specify the following:

- Beginning-of-tape (BOT)
- End-of-tape (EOT)
- End-of-file (EOF)
- Corrected errors (CER)
- Uncorrected or hard errors (HER)

The format of the geometry dump follows:

```
FILE#       RECORD#          LENGTH        STATUS
BOT          BEGINNING OF TAPE              BOT
 ffff          rrrr           bbbb        status
  .             .              .             .
  .             .              .             .
  .             .              .             .
 ffff          rrrr           bbbb          EOF
EOT             END OF TAPE                  EOT
```

ffff	specifies the number of file(s) being dumped.
rrrr	specifies the number of records being dumped within file ffff.
bbbb	specifies the length in bytes of record rrrr.
status	specifies End-of-File (EOF), corrected errors (CER) or hard errors (HER).

If the RR(n) argument is specified, the data within the record(s) specified is dumped in a side-by-side hexadecimal ASCII format following the line of geometry (file, record, length, and status) for that record.

5.3.5 "E" Directive

..,ER,..
..,ER(n),..
..,ER='CC',..

Inserts an erase I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic performs the number of erases specified by n, or if n is not specified, the number of erases specified by parameter Y. If Command Chain 'CC' is specified, IOCD word 2, bit 1, is set and the next directive is processed.

5.3.6 "T" Directive

..,INCH,..
..,INCH='CC',..

If Command Chain 'CC' is not specified, the diagnostic performs an immediate Initialize Channel (INCH) command by executing a Start I/O (SIO) instruction to an INCH command that is its own I/O command list (IOCL). If Command Chain 'CC' is specified, the diagnostic places an INCH I/O command doubleword (IOCD) into the I/O command list (IOCL) queue, with bit 1 of IOCD word 2 set, and then processes the next directive.

The INCH command must be the first operation performed on a channel that has been reset by either a System Reset from the control panel or by a Reset Channel ..,RCH,.. directive. If a System Reset is executed, the diagnostic automatically performs an INCH command in its initialization routine.

5.3.7 "L" Directives

..,LOOP,..

Establishes a Loop on Error point within a directive string entered from the keyboard. Option 3 (Repeats the sequence detecting an error) must be set for the looping function to operate. This directive is used if a string of operator-composed directives is being processed. The Loop on Error occurs from the last place of usage in the operator-composed directive string. If more than one Loop on Error is in the operator-composed directive string, the ..,LOOP,.. directive previous to the error condition is used as the starting point.

..,LSC,..

This directive is used by the Scatter/Gather test to prepare the output buffer for a data compare prior to reading data from the drive in the Scatter/Gather mode. However, the following procedure is needed to obtain this data:

1. The output data has two images. The first image starts at the beginning of the input buffer. The second image of the same data is prepared by setting parameter L and preparing the second buffer with the same data.

   Example

   PW=V,PW/2,PL=0,BCO,LBP,PL=W,BCO,LBP

   Parameter L now points to the second image in the upper output buffer.

2. The ..,LSC,.. directive takes the value of parameter L and the current bias and constructs the data in the lower output buffer. This data in the lower buffer has data from the upper buffer and zero data words which are dependent on the Scatter/Gather bias.

..,LSG,..

This directive is used by the Scatter/Gather test to prepare the output buffer for a data compare after data is written to the drive. However, the following procedure is needed to obtain this data:

1. The output data has two images. The first image starts at the beginning of the output buffer. The second image of the same data is found by previously setting parameter L and preparing the second buffer with the same data.

   Example

   PW=V,PW/2,PL=0,BCO,LBP,PL=W,BCO,LBP

   Parameter L now points to the second output buffer.

2. The ..,LSG,.. directive takes the value in parameter L and the current bias and constructs the data in the lower output buffer (which has the original output address) with the data that is expected to be input with the Read commands.

5.3.8 "N" Directive

..,NOP,..
..,NOP='CC',..

Inserts a No Operation (NOP) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain is specified, IOCD word 2, bit 1, is set and the next directive is processed.

5.3.9 "R" Directives

..,R,..
..,Rn,..
..,R='arg_1,arg_2,...,arg_n'

Inserts a Read I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by n, or if n is not specified, the value specified in parameter W as the byte count.

Legal arguments for this directive are: DC, CC, SLI, SKIP, BC(#), NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

..,RB,..
..,RBn,..
..,RB='arg_1,arg_2,...,arg_n',..

Inserts a Read Backward I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by n, or if n is not specified, the value specified in parameter W as the byte count.

Legal arguments for this directive are: DC, CC, SLI, SKIP, BC(n), NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

..,RC,..
..,RCn,..
..,RC='arg_1,arg_2,...,arg_n',..

Inserts a Read Compare I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by n, or if n is not specified, the value specified in parameter W as the byte count.

Legal arguments for this directive are: DC, CC, SLI, SKIP, BC(n), NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

..,RCH,..

Issues a command device (CD) Reset Channel instruction.

The Reset Channel directive must be followed by an Initialize Channel ..,INCH,.. directive to avoid problems with interrupt status reporting on subsequent operations.

..,RR,..
..,RRn,..

Inserts a Read Record I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by parameter W. If parameter W equals zero, or if less than eight bytes are specified, eight bytes are read. The diagnostic then issues the number of Command Device (CD) Start I/Os specified by n, or if n is not specified, the value specified in parameter R is used as the number of CD Start I/Os to issue.

A complete status and data comparison is performed following each Start I/O. This directive is convenient for reading multiple records of a fixed byte count.

..,RRB,..
..,RRBn,..

Inserts a Read Record Backward I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by parameter W. If parameter W equals zero or if less than two bytes are specified, two bytes are read. The diagnostic then issues the number of Command Device (CD) Start I/Os specified by n, or if n is not specified, the value specified in parameter R is used as the number of (CD) Start I/Os to issue.

This directive is convenient for reading backward multiple records of a fixed byte count. A complete status and data comparison is performed following each record read.

..,RRC,..
..,RRCn,..

Inserts a Read Compare I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic reads the number of bytes specified by parameter W. If parameter W equals zero, or if less than two bytes are specified, two bytes are read. The diagnostic then issues the number of Command Device (CD) Start

I/O's specified by n, or if n is not specified, the value specified in parameter R is used as the number of CD Start I/Os which are issued.

..,RW,..
..,RW='arg_1,arg_2,...,arg_n',..

Inserts a Rewind I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, the next IOCD is queued by the channel and executed after the device reaches beginning-of-tape (BOT).

Legal arguments for this directives are: CC, NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

..,RWU,..

Places a Rewind and Unload I/O command doubleword (IOCD) into the I/O command list (IOCL) and performs an immediate Start I/O instruction. This command is drive-dependent in that a 75 ips drive responds by rewinding and going off-line, while a 45 ips drive goes off-line without rewinding.

..,RWW,..

Inserts a Rewind and Wait I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic waits for the device to reach beginning-of-tape (BOT) before processing the next directive.

5.3.10 "S" Directives

..,SIN,..

Issues a Command Device (CD) Start I/O instruction. This directive is used to repeat a command sequence previously entered or to initiate an operation created by a directive sequence that terminates with ..,NSIO,.. . This directive issues the Start I/O instruction but does not wait for an interrupt. This directive is used in conjunction with Halt I/O, Stop I/O, and Test I/O to determine the condition code detected after the Start I/O command and the next extended I/O command. The directive ..,DCC,.. , which writes the last condition code received to the listed output (LO) file, may be used with this directive.

An example of this directive and the list condition code directive to check condition codes is shown below:

```
..,1,WRI,J1/500,RWW='NSIO',SIN,TIO,DCC,TIO,..
```

..,SIO,..

Issues a Command Device (CD) Start I/O instruction. This directive is used to repeat a command sequence previously entered or to initiate an operation created by a directive sequence that terminates with ..,NSIO,.. . This directive waits for an interrupt to be received from the device.

..,SM='arg_1,arg_2,...,arg_n',..

Inserts a Set Mode Register I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. This directive is required in selecting the desired operating characteristics for a tape drive.

For a description of the optional specifications, refer to Mode Register Arguments.

..,SNS,..
..,SNS='arg_1,arg_2,...,arg_n',..

Inserts a Sense I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. This directive is used to acquire the Sense status from a device prior to displaying the Status with a ..,DSD,.. directive.

Legal arguments for this directive are: CC, SLI, NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

5.3.11 "T" Directives

..,TICn,..

Inserts a Transfer-in-Channel (TIC) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If the TIC is executed, control transfers to the IOCD in the IOCL queue slot number specified by n. A Transfer-in-Channel is an implied command chain operation because the flag byte of the previous IOCD is passed through the Transfer-In-Channel. Therefore, a Transfer-in-Channel may never be the first IOCD in the IOCL queue, nor can a TIC reference another TIC.

..,TIO,..

Issues a Command Device (CD) Test I/O instruction.

5.3.12 "W" Directives

..,W,..
..,Wn,..
..,W='arg_1,arg_2,...,arg_n',..

Inserts a Write I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic writes the number of bytes specified by n, or if n is not specified, the value specified in parameter W as the byte count.

Legal arguments for this directive are: DC, CC, BC(c), NSIO

For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

..,WFMn,..
..,WFM='CC',..

Inserts a Write File Mark I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic writes the number of file marks specified by n, or if n

is not specified, the number of file marks specified by parameter Y. If parameter Y equals zero, one file mark is written.

If Command Chain 'CC' is specified, IOCD word 2, bit 1 is set, and the diagnostic processes the next directive. Otherwise, a command device (CD) Start I/O instruction is issued. A complete status verification is performed following each file mark written.

..,WR,..
..,WRn,..

Inserts a Write Record I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The diagnostic writes the number of bytes specified by parameter W. If parameter W equals zero, or if less than two bytes are specified, two bytes are written. The diagnostic issues the number of Command Device (CD) Start I/Os specified by n, or if n is not specified, the value specified in parameter R as the number of Start I/Os to issue. If parameter R equals zero, one Start I/O is issued. A complete status verification is performed following each Start I/O.

This directive is convenient for writing multiple records of a fixed byte count.

5.4 Directives Issuing Start I/O Instructions

Directives that do not specify Command Chain (CC), Data Chain (DC), or No Start I/O (NSIO) cause the diagnostic to issue an immediate Start I/O instruction once the I/O command doubleword (IOCD) is inserted in the I/O command list (IOCL) queue. Directives that do specify Command Chain (CC), Data Chain (DC), or No Start I/O (NSIO) cause the diagnostic to insert an I/O command doubleword (IOCD) in the I/O command list (IOCL) queue and then process the next directive without issuing a Start I/O (SIO).

5.5 Directive Arguments

Arguments are appended to specified directives. The argument syntax follows:

..,directive='arg_1,arg_2,...,arg_n',..

There are four categories of arguments:

- IOCD Flag Byte Arguments
- IOCD Byte Count Arguments
- Mode Register Arguments
- Dump Directive Arguments

Descriptions of the four categories follow.

5.5.1 IOCD Flag Byte Arguments

The I/O command doubleword (IOCD) flag byte arguments specify modifications made to an individual IOCD within the I/O command list (IOCL) queue. IOCD flag byte arguments are specified in any order. Consult individual directive derivatives to define valid IOCD flag byte arguments. The IOCD flag byte arguments are defined as follows:

DC Data Chain

Causes the diagnostic to set IOCD word 2, bit 0, indicating that if the byte count equals zero, proceed with the next IOCD in the IOCL queue. The next IOCD must be the same type of command as the data chained command.

CC Command Chain

Causes the diagnostic to set IOCD word 2, bit 1, indicating that at end-of-record, proceed with the next IOCD in the IOCL queue. The next IOCD may be any valid IOCD. If end-of-record is reached prior to byte count equals zero or if byte count equals zero is reached prior to the end of record, an incorrect length status is reported and IOCL execution is terminated.

SLI Suppress Incorrect Length Indication

Causes the diagnostic to set IOCD word 2, bit 2, indicating that if end-of-record is reached prior to the end-of-record, the incorrect length status normally reported is to be suppressed and IOCL execution is to continue.

SKIP Skip

Causes the diagnostic to set IOCD word 2, bit 3, indicating that the number of bytes specified are to be skipped (not written to memory). This flag is only valid for Read, Read Backward, and Read Compare IOCDs.

PPCI Post Programmed Controlled Interrupt

Causes the diagnostic to set IOCD word 2, bit 4, indicating that status is to be posted and a service interrupt (SI) is to be generated upon completion of the currently executing IOCD (at byte count equals zero if data chaining is specified or at Device End if command chaining is specified).

SYC Sync Chase

Causes the diagnostic to set I/O command doubleword (IOCD) word 2, bit 7, indicating that the command operates in the Sync Chase mode, depending upon what parameters are used if the Set Mode Register command is executed. This flag is valid only for Read, Read Backward, and Read Compare IOCDs.

5.5.2 IOCD Byte Count Argument

The I/O command doubleword (IOCD) byte count argument specifies modifications made to an individual IOCD within the I/O command list (IOCL) queue. The IOCD byte count argument is specified along with IOCD flag byte arguments. Consult individual directive derivatives to define valid byte counts.

BC(n)

Conditions IOCD word 2, bits 16 through 31, to contain the value specified by (n). The minimum byte count allowed for Read, Read Backward, Read Compare, and Write IOCDs is two bytes.

5.5.3 Mode Register Arguments

The Mode Register in the High-Speed Tape Processor contains 11 bytes that have the following configuration:

Mode Byte 0

Bit 0	State=0	Write Data Chained commands are broken if a corrected error is detected.
	State=1	Write Data Chained commands are not broken if a corrected error is detected, provided the error is only on one track.
Bit 1	State=0	If bit 0 and bit 6 of the mode register are zero, the drive operates in the NRZI mode.
	State=1	Drive operates in the PE mode.
Bit 2	State=0	Write Data Chained commands are broken if end-of-tape (EOT) is detected.
	State=1	Write Data Chained commands are not broken if end-of-tape (EOT) is detected.
Bits 3-5		Not used.
Bit 6	State=0	Drive is not in GCR mode.
	State=1	Drive operates in GCR mode.
Bit 7	State=0	Drive does not operate in Scatter/Gather mode.
	State=1	Drive operates in Scatter/Gather mode.

Mode Bytes 1 and 2

This field contains the Scatter/Gather bias used by write/read operations if the drive is operating the Scatter/Gather mode.

Mode Byte 3

This field contains the Sync Chase byte count used if read operations are in the Sync Chase mode.

Mode Bytes 4 through 7

This field contains the Sync Chase Word used by read operations in the sync chase mode.

Mode Bytes 8 through 11

This field contains the Sync Chase mask word used by read operations in the Sync Chase mode.

Following a system reset, the mode register format arguments are set to GCR.

The following are the arguments and the associated data bytes used by the High-Speed Tape Processor:

..,ZERO $\left\{\begin{matrix}(0)\\(1)\end{matrix}\right\}$,..

ZERO (1) causes Write Data Chained commands not to be broken if a corrected error is detected. ZERO (0) causes Write Data Chained commands to be broken if a corrected error is detected along with a hard error sense bit setting and a status error of data check.

..,NRZI,..

Causes mode bit 1 and the GCR bit 6 to be reset. The expected sense bit 27 is set and the GCR expected sense bit 32 is reset.

..,PE,..

Causes mode bit 1 to be set and resets the GCR bit 6 in the mode register. The expected sense bits 27 and 31 are reset.

..,TWO $\left\{\begin{matrix}(0)\\(1)\end{matrix}\right\}$,..

TWO (1) causes Write Data Chained commands not to be broken if end-of-tape (EOT) is detected. TWO (0) causes Write Data Chained commands to be broken if EOT is detected.

..,GCR,..

Cause mode bit 6 to be set and resets mode bit 1. The expected sense bit 27 is reset and bit 31 is set.

..,SGM,..

Sets bit 7 of the mode register to operate in the Scatter/Gather mode.

..,RSGM,..

Resets bit 7 of the mode register, which enables the drive to operate in the normal mode.

Scatter Gather Bias (Bytes One and Two)

..,SGB(n),..

Sets the Scatter/Gather Bias specified by (n) into the Scatter/Gather bytes 1 and 2.

..,SGBC,..

Sets the Scatter/Gather Bias specified by parameter 0 into the Scatter/Gather bias bytes 1 and 2. This argument, used in a directive string, enables the string to use a variable bias controlled by parameter 0.

..,SWB(n),..

Sets the Sync Chase byte count bytes 1 and 2 to the number specified by (n).

Sync Chase Word (Bytes 4 through 7)

..,SW(n),..

Causes the hexadecimal value specified by (n) to be inserted into the Sync Chase Field (bytes 4 through 7). In the sync chase mode of operation, this is the data pattern that the Read commands will use as a sync.

..,SMF,..

Causes the data value in parameter F to be added to the starting address of the output buffer and retrieves four bytes of data from that location. This data is used as the Sync Chase Word. The output buffer address then is set to the location past the four bytes extracted from the data. Effectively, the output buffer starting address now points to the location past the Sync Chase Word and data compares should function properly on Read Sync commands.

Sync Mask Word (Bytes 8-11)

..,MW(n),..

Causes the hexadecimal value specified by (n) to be inserted into the Sync Mask Word Field (bytes 8 through 11). In the sync chase mode of operation, this is the data pattern that the Read commands uses as a mask.

Example

A sync mask word of FFFFFF00 enables the High-Speed Tape Processor to search for the first three bytes of data and ignore the rest.

..,MWC,..

Causes the data value in parameter C to be inserted into the Sync Mask Word field (bytes 8 through 11). This argument, used in a directive string, enables the string to use a variable mask word pattern controlled by parameter C.

Command Chain Flag

..,CC,..

Causes IOCD word 2, bit 1 to be set and the diagnostic processes the next directive. Otherwise, a Command Device (CD) Start I/O instruction is issued. A complete status verification occurs following the Set Mode register operation.

5.5.4 DUMP Directive Arguments

The ..,DUMP,.. directive arguments specify the functions performed to position a tape and the information on that tape to be dumped (geometry/data). The arguments are defined as follows:

RW

Performs a Rewind and Wait (RWW) I/O command doubleword (IOCD), waiting for the tape drive to reach beginning-of-tape (BOT) before processing the next directive argument. If specified, this should be the first argument in the directive string.

GEOM

Initializes the record/file label counters to one, then performs Read Record I/O command words (IOCDs) until end-of-tape is reached. The Read Record IOCD is equivalent to the ..,R='BC(4096), SKIP',.. directive. For each Read Record IOCD performed, the Sense/Status data is checked and the file number, record number, record length (in bytes), and status (either BOT, EOF, EOT, CER or HER) is written to the listed output (LO) file. If the Rewind (RW) argument does not precede the GEOM argument, the geometry scan starts at the current tape position, but the file/record numbers are initialized to one.

RR(n)

Performs a Read Record I/O command doubleword (IOCD) for the number of records specified by (n) and writes the data for each record (a maximim of 4096 bytes per record), preceded by its geometry, to the listed output (LO) file. The number of records to be read (n) must be a value within the range 1 through 255 and must be enclosed in parentheses. If a file boundary is crossed, the file label counter is incremented by one and the record label counter is reset to one.

AR(n)

Performs an Advance Record I/O command doubleword (IOCD) for the number of records specified by (n) and writes one line of geometry to the listed output (LO) file for each record and file mark processed. The number (n) of Advance Records to be performed must be a value within the range one through 255 and must be enclosed in parentheses. If a file boundary is crossed, the file label counter is incremented by one and the record label counter is reset to one.

BR(n)

Performs a Backspace Record for the number of records specified by (n) and writes one line of geometry to the listed output (LO) file for each record and file mark processed. The number of Backspace Records to be performed (n) must be a value within the range one through 255 and must be enclosed in parentheses. If a file boundary is crossed, the file label counter is decremented by one and the record label counter becomes meaningless. Therefore, the process is halted with the tape positioned ahead of the EOF mark and only the number of records processed prior to encountering the EOF is reflected in the geometry written to the LO file.

AF(n)

Performs an Advance File for the number of files specified by (n) and writes one line of geometry to the listed output (LO) file for each file processed. The number of Advance Files to be performed (n) must be a value within the range one through 255 and must be enclosed in parentheses. For each file boundary crossed, the file label counter is incremented by one and the record label counter is reset to one, but individual record geometry is not reported.

BF(n)

Performs a Backspace File for the number of files specified by (n) and writes one line of geometry to the listed output (LO) file for each file processed. The number of Backspace Files to be performed (n) must be a value within the range one through 255 and must be enclosed in parentheses. For each file boundary crossed, the file label counter is decremented by one and the record label counter becomes meaningless. No record geometry is reported.

5.6 Directive Error Reporting

The following messages are written to the listed output (LO) file if an operator-composed directive violates a protocol rule for magnetic tape extended I/O:

If more than 16 IOCDs are specified to be written into the IOCL queue, the following message is written:

NUMBER OF DIRECTIVES PROCESSED WOULD HAVE CAUSED THE IOCD LIST TO OVERFLOW

If an attempt is made to data chain a specific command to a different command, the following message is written:

CANNOT DATA CHAIN TO A DIFFERENT IO COMMAND

If the last IOCD in the IOCL queue is data chained and that no terminal IOCD is specified, the following message is written:

DIRECTIVE STRING ENDED WITH DATA CHAIN SET

If the data chained byte count exceeds the maximum size of the output buffer, the following message is written:

DATA CHAIN WOULD HAVE CAUSED OVERRUN OF OUTPUT BUFFER

If the program detects a Transfer-in-Channel (TIC) to a Transfer-in-Channel (TIC) command, the following message is written:

TIC TO TIC IS INVALID

This sequence would cause an IOCD validation error.

If a Transfer-in-Channel (TIC) command is the first command entered into the IOCD queue, the following message is written:

TRANSFER IN CHANNEL AS FIRST IOCD INVALID

If a Transfer-in-Channel (TIC) command is pointing to a forward IOCD in the queue but the slot is empty, the following message is written:

TIC COMMAND POINTS TO UNFILLED ENTRY IN IOCD QUEUE

If a subaddress greater than three is assigned to the device, the following message is written:

SA * IS ILLEGAL (VALID SUBADDRESSES ARE 0-3)

* specifies the invalid subaddress assigned to the device.

The valid subaddresses for the High-Speed Tape Processor are zero through three.

5.7 Parameter Descriptions

The High-Speed Tape Processor Diagnostic uses the following set of 26 unique parameters (identified A through Z) that are manipulated by the P directive processor:

Parameter	Use	Description
A	Read Only	Used by the Diagnostic Executive Program (DEXP).
B	Read/Write	Available for general use.
C	Read/Write	Used by Sync Chase.
D	Read/Write	Contains the current output buffer data pattern if the output buffer is set using a ..,PD( )k,.. directive.
E	Read/Write	Available for general use.
F	Read/Write	Used by Sync Chase.
G	Read/Write	Available for general use.
H	Read/Write	Available for general use.
I	Read/Write	Contains the current output buffer word one data pattern if the output buffer word one is set using a ..,PI( )K,.. directive.
J	Read/Write	Available for general use.
K	Read/Write	Used by ..,BCI,.. directive.
L	Read/Write	Used by directives ..,LSC,.. and ..,LSG,.. to determine address of second output buffer. Also used by ..,BCO,.. directive.
M	Read/Write	Available for general use.
N	Read/Write	Available for general use.
O	Read/Write	Used by Scatter/Gather for variable bias.
P	Read/Write	Available for general use.
Q	Read/Write	Available for general use.
R	Read/Write	Contains the default record count for ..,WR,.. and ..,RR,.. directives.
S	Read/Write	Available for general use.
T	Read/Write	Available for general use.
U	Read/Write	Available for general use.
V	Read Only	Contains the maximum transfer count (4096 bytes).
W	Read/Write	Contains the transfer count for ..,W,.., ..,R,.., ..,WR,.., and ..,RR,.. directives. Default is 4096.
X	Read/Write	Contains the implied hold duration for ..,H,.. and ..,HS,.. directives.
Y	Read/Write	Contains the implied iteration count for ..,ER,.., ..,BR,.., ..,AR,.., ..,BF,.., and ..,AF,.. directives.
Z	Read/Write	Contains an indirect address for ..,JZ,.. directives.

Gould RPU Disc Processor Media Verification Program

(RP.MVP.r)

Disc Utility Description

Diagnostic Number: 001032-905r

Documentation Release Level: 1.1

Documentation Release Date: November 1984

Publication Order Number: 326-005320-100

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

(C) Copyright 1984
Gould Inc., Computer Systems Division
All Rights Reserved
Printed in U.S.A.

HISTORY

The Gould RPU Disc Processor Media Verification Program (RP.MVP.r) Disc Utility Description, Publication Order Number **326-005320-100,** was printed November 1984.

This manual contains the following pages:

Title page
Copyright page
History page
iii through iv
1-1/1-2
2-1 through 2-6
3-1 through 3-6
4-1 through 4-16
5-1 through 5-37/5-38
6-1/6-2

CONTENTS

Chapter Page

1 DESCRIPTION AND REQUIREMENTS

1.1 General Description 1-1
1.2 Resource Requirements 1-1

2 ASSIGNMENTS AND INITIALIZATION

2.1 Disc Processor Channel Model Number Assignment 2-1
2.2 System Disc Flag Assignment 2-1
2.3 Formatted Drive Assignment 2-1
2.4 Port Assignments 2-1
2.5 Utility Initialization 2-2
2.6 Utility Option Set 2-5
2.7 Utility Execution 2-6

3 TEST DESCRIPTION

3.1 Test Summary 3-1
3.2 Disc Format, Label Verification, and Media Verification Test 3-1
3.2.1 Label Verification 3-1
3.2.2 Format and Label Verification 3-2
3.2.3 Media Verification 3-4

4 ERROR REPORTING

4.1 Error Message Assignment 4-1
4.2 Error Messages and Message Sequence 4-1
4.3 Status Error Report Exceptions 4-4
4.4 Condition Code Errors 4-5
4.5 No Interrupt and Device End Messages 4-5
4.6 Status Address Word Error 4-6
4.7 Program Start Errors 4-7
4.8 Additional Messages 4-8
4.9 Directive Error Reporting 4-9
4.10 Label Verification Messages 4-11
4.10.1 Description of Label Flag Byte One 4-12
4.10.2 Description of Label Flag Byte Two 4-13
4.11 Track Deallocation Messages 4-13

Chapter Page

5 DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Language Compiler 5-1
5.2 DEXP Standard Directives 5-1
5.2.1 "A" Directives 5-1
5.2.2 "C" Directives 5-2
5.2.3 "D" Directives 5-2
5.2.4 "H" Directives 5-3
5.2.5 "J" Directives 5-3
5.2.6 "L" Directives 5-4
5.2.7 "P" Directives 5-6
5.2.8 "S" Directives 5-7
5.2.9 "T" Directives 5-8
5.3 Gould RPU Disc Processor Media Verification Program Unique Directives 5-8
5.3.1 "A" Directives 5-8
5.3.2 "B" Directives 5-9
5.3.3 "C" Directives 5-10
5.3.4 "D" Directives 5-12
5.3.5 "F" Directives 5-17
5.3.6 "H" Directive 5-18
5.3.7 "I" Directives 5-18
5.3.8 "L" Directives 5-18
5.3.9 "M" Directive 5-20
5.3.10 "N" Directive 5-21
5.3.11 "R" Directives 5-21
5.3.12 "S" Directives 5-23
5.3.13 "T" Directives 5-25
5.3.14 "W" Directives 5-28
5.3.15 "X" Directive 5-28
5.4 Directives Issuing Start I/O Instructions 5-29
5.5 Directive Arguments 5-29
5.5.1 IOCD Flag Byte Arguments 5-29
5.5.2 IOCD Byte Count Arguments 5-30
5.5.3 Test STAR Specification Arguments 5-30
5.5.4 Load Mode Register Arguments 5-31
5.5.5 Track and Sector Flag Byte Arguments 5-32
5.5.6 Access Control Flag Byte Arguments 5-34
5.6 Parameter Descriptions 5-34
5.7 Disc Drive Test Parameters 5-36
5.8 Model Number Drive Test Parameters 5-37

6 I/O COMMAND TRACE

6.1 Tracing I/O Activity 6-1
6.2 I/O Command Trace 6-1

CHAPTER 1

DESCRIPTION AND REQUIREMENTS

1.1 General Description

The Gould RPU Disc Processor Media Verification Program (RP.MVP.r) is a Level Two program which loads and executes under control of the Diagnostic Executive Program (DEXP).

This utility program formats and verifies media and produces a Deallocated Track Table. It does not set up alternate and defective tracks or manipulate Deallocated Track Table information. However, this table may be accessed to develop specific system requirements.

The utility program requires a minimum of operator intervention.

During execution the utility program writes milestone messages and error message sequences to the listed output (LO) file which is assigned to the operator's console or a line printer.

1.2 Resource Requirements

The Gould RPU Disc Processor Media Verification Program requires:

- Gould CPU with a minimum of 64K words of memory.
- A Disc Processor (Model 8055).
- A Universal Disc Processor (Model 8060), or
- A Cache Disc Accelerator (Model 8160, 8161, 8162, 8180, 8181, or 8182).
- One or more of the following disc drives:

Fixed Head Disc Drive (5 MB)	Model 8104
Cartridge Module Disc Drive (32 MB)	Model 8117
Winchester Drive (80 MB)	Model 8122
Winchester Drive (160 MB)	Model 8127
Nine Inch Cartridge Disc RSD (80 MB)	Model 8136
Nine Inch Winchester Disc FSD (160 MB)	Model 8146
Winchester Disc Drive (675 MB)	Model 8155
Eight Inch Winchester Drive (80 MB)	Model 8420
Eight Inch Winchester Drive (167 MB)	Model 8424
Nine Inch Winchester Disc (340 MB)	Model 8858
Removable Disc Drive (80 MB)	Model 9342
Removable Disc Drive (300 MB)	Model 9344

Note: The Model 8155 (675 MB) disc is analogous to the Model 8155 (600 MB) disc.

CHAPTER 2

ASSIGNMENTS AND INITIALIZATION

2.1 Disc Processor Channel Model Number Assignment

The Gould RPU Media Verification Program defaults to the two-board Disc Processor Model 8055. For Disc Processor Model 8060, the ..,CON,.. directive should be used as follows:

```
DXP>taskname CON8060
```

2.2 System Disc Flag Assignment

The System Disc Flag defaults to a setting of ON, which enables the Gould RPU Media Verification Program to perform only Label Verification of the entire media. This allows the utility to read labels on previously formatted packs. Information is not written to the media in this mode of operation. If the System Disc Flag is OFF, the media is formatted, and if option 19 is reset, label verification and deallocated track information are written to the media. If option 19 is set with the System Disc Flag OFF, the entire media is formatted, verified, and deallocated tracks are written to the media. Refer to the ..,SDF,..Unique Directive in Chapter 5.

If one or more drives are used to format the media, the System Disc Flag is set ON or OFF for each drive.

Note: The System Disc Flag Directive ..,SDF,.. always precedes the Dual Port directive ..,DUAL0,.. and the Single Port directive ..,SING,.. in a directive string.

2.3 Formatted Drive Assignment

The utility program defaults to a single port drive attribute. If the media is formatted to dual port, refer to Chapter 5 for a complete description of the ,..DUAL0,.. Unique Directive.

2.4 Port Assignments

The utility program defaults to the single port, 20-sector format. However if the media is formatted in the dual port mode or 16-sector format, the media must be formatted as follows:

	Operating Procedure	Example
1.	Load the Diagnostic Executive Program (DEXP).	LOAD DEXP
2.	Load the Media Verification Program (MVP).	LOAD D1 (RP.MVP.r)

Operating Procedure	Example
3. Perform the initialization sequence to assign device addresses.	See Utility Initialization Sequence
4. Set the System Disc Flag 'OFF' and establish Dual Port attributes and parameters for the media.	D1 SDF= 'SYS(OFF)', DUAL0
5. Reset the repeat option flag.	RESET D1,OPT,6
6. Set the Media Verification option.	SET D1,OPT,19
7. Start the Program.	START D1
8. Upon completion of formatting and verifying the media, the program will output to the 'LO' file a list of deallocated tracks.	
Note: For a 16-sector format, Step 4 would be modified as follows:	D1 FOR16,SDF='SYS(OFF)',DUAL0

2.5 Utility Initialization

Before initializing the Gould RPU Disc Processor Media Verification Utility, load the program into the system by following the procedures in Chapter 7--Level Two Diagnostics, of the Gould Virtual Memory Diagnostic Facility User's Manual. The program name for loading the utility is RP.MVP.r. After the utility finishes loading, it writes the following initialization sequence to the operator communications (OC) file:

Note: Entering an invalid response to the prompts causes the diagnostic to reissue them.

1. The utility writes the following message:

   'taskname' INITIALIZATION

2. The utility then issues the following prompt:

   SELECTED CHANNEL ADDRESS=08; [CR,?,X,##]?

   Responding with a carriage return (cr) causes the utility to assign the specified default channel address (08). To modify the default channel address, enter the valid two-character hexadecimal channel address (##). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

   The valid channel address range is 00_{16} through $7F_{16}$.

3. The utility then issues the following prompt:

```
SELECTED NUMBER OF DRIVES=1; [CR,?,X,#]?
```

Responding with a carriage return (cr) causes the utility to assign the specified default number of drives (1). To modify the default number of drives, enter the valid one-character decimal number of drives (#). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

The valid number of drives is from one to eight.

4. The utility then issues the following prompt:

```
SUBCHANNEL INITIALIZATION
SELECTED SUBCHANNEL ADDRESS=00; [CR,?,X,#]?
```

Responding with a carriage return (cr) causes the utility to assign the specified default subchannel address (00). To modify the default subchannel address, enter the valid two-character hexadecimal subchannel address (#). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

The valid subchannel addresses are all even values 0_{16} through E_{16}.

5. The utility then issues the following prompt:

```
SELECTED MODEL NUMBER=9342; [CR,?,X,####]?

ENTRY OF A MODEL NUMBER WILL SELECT THE DRIVE TO SINGLE
PORT
```

Responding with a carriage return (cr) causes the utility to assign the specified default model number (9342). To modify the default model number, enter the valid four-character model number (####). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

Valid model numbers are as follows:

Model #	Cylinder	Heads	Size		Description
8104	64	4	5 MB		Fixed Head
8117	823	2	32 MB	14-in.	Cartridge
8122	823	5	80 MB	14-in.	Winchester
8127	823	10	160 MB	14-in.	Winchester
8136	823	5	80 MB	9-in.	Cartridge
8146	823	10	160 MB	9-in.	Winchester
8155	843	40	675 MB	14-in.	Winchester
8420	823	5	80 MB	8-in.	Winchester
8424	1024	8	167 MB	8-in.	Winchester
8858	711	24	340 MB	9-in.	Winchester
9342	823	5	80 MB	14-in.	Removable
9344	823	19	300 MB	14-in.	Removable

6. The utility then issues the following prompt:

   SELECTED SECTOR FORMAT=20; [CR,?,X,##]?

   Responding with a carriage return (cr) causes the utility to assign the specified default sector format (20). To modify the default sector format, enter the valid two-character sector format (##). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

   Valid sector formats are 16 and 20 sectors per track.

   If the model number is changed during reinitialization, the sector format default is reset to 20.

7. The utility then issues the following prompt:

   SELECTED PORT CONDITION=SING; [CR,?,X,TTTT#]??

   Responding with a carriage return (cr) causes the utility to assign the specified default port condition (SING). To modify the default port condition, enter the valid port condition (TTTT#). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

   The following are valid port conditions:

   SING — Entire functional surface or two utility cylinders are tested using one disc drive in a single system.

   DUAL0 — Entire functional surface or two utility cylinders are tested using one disc drive shared by two systems.

   If the model number is changed during reinitialization, the selected port condition default is reset to SING (single port).

8. The diagnostic then issues the following prompt:

   IS CACHE DISC ACCELERATOR PRESENT?=N [CR,?,X,Y, N]

   Responding with a carriage return (cr) causes the utility to assign the specified default Cache Disc Accelerator Condition (N). To modify the default Cache Disc Accelerator Condition, enter the valid Cache Disc Accelerator Condition (Y). Responding with a question mark (?) causes the utility to write the range of values to be entered. Responding with an X, exits the initialization sequence before its completion.

9. The Utility then writes the following message:

   USE HELP DIRECTIVE FOR OTHER INFORMATION
   INITIALIZATION COMPLETE

The first message is only for information. The second message indicates utility initialization is successful. At this time, modify options and/or start the utility program.

After initialization or reinitialization, the values last entered during the initialization or reinitialization sequence are the default values for the next initialization or reinitialization.

However, if a model number is changed during reinitialization, the utility does not follow the "last initialization value entered" rule.

2.6 Utility Option Set

After the Gould RPU Media Verification Program loads and initializes, it enters a wait state until receiving directions through the Gould Diagnostic Executive Program (DEXP).

Using the DEXP, select options to modify default execution conditions.

The Gould RPU Media Verification Program contains the following options:

Option	Default	Effect When Set
0	Reset	Inhibits all attempted error messages. If the option is set during an error message report, the message is completed. Option 1 is not inhibited by this option.
1	Reset	Rings the bell on the operator's console once per error.
2	Reset	Suspends execution if an error is detected. Prior to suspension, the associated error message is written, unless option 0 is set. During suspension, the following message is written: 'taskname' ERROR. . . ; [C,H,T] ? The query [C,H,T]? requires a response. Responding with C, continues execution. Responding with T, terminates execution. Responding with H, places the diagnostic on hold, allowing time for setting options, performing memory dumps, or manipulating other diagnostics running concurrently. Nullify the hold state with a Continue command.
3	Reset	Unimplemented.
4	Reset	Unimplemented.
5	Reset	Unimplemented.
6	Reset	Unimplemented.
7	Reset	Unimplemented.
8	Reset	Unimplemented.
9	Reset	Inhibits condition code errors and no status interrupt received messages.
10	Set	Writes the utility program's pass count message to the listed output (LO) file.

Option	Default	Effect When Set
11	Reset	Writes the utility program's milestone messages to the listed output (LO) file.
12	Reset	Inhibits both Status and Sense data error message expansion.
13	Reset	Inhibits Status error messages.
14	Reset	Inhibits Sense error messages.
15	Reset	Inhibits Data error messages.
17	Reset	Enables I/O Command Trace.
19	Reset	Conditions the utility program to operate in the Media Verification Mode if Test 1 is performed. The System Disc Flag must be set to OFF to enable the utility program to check this option. This option is not checked if the System Disc flag is ON.

2.7 Utility Execution

After setting options, issue a Start command to begin diagnostic execution. As the diagnostic begins execution, it writes the following message to the operator communications (OC) file:

```
START taskname (#001032-905r date)
(C) COPYRIGHT year GOULD INC., COMPUTER SYSTEMS DIVISION
ALL RIGHTS RESERVED
```

Once execution starts, the diagnostic begins testing and writes milestone messages to the listed output (LO) file.

CHAPTER 3

TEST DESCRIPTION

3.1 Test Summary

The Gould RPU Disc Processor Media Verification Program has the capability of formatting media and testing the media to determine the coordinates of defective tracks and to identify these tracks in a Deallocation Track Table. The Deallocation Track Table contains the coordinates of all defective tracks, the coordinates of the diagnostic cylinders, the coordinates of the Deallocation Track Table, and the hardware test track.

After verifying the media and establishing the deallocated tracks, the program inserts the binary sector address of the deallocation track information into bytes 12 through 16 of track label zero, head zero. The pointer in track label zero, head zero is used by the Gould RPU Disc Processor Diagnostic and by the designated operating system.

Additional tracks are added to the Deallocation Track Table by using the ..,DT,.. directive. This directive is used if false address marks are included on a media flaw map but are not detected by the media verification program, therefore, using the directive includes them in the Deallocation Track Table.

3.2 Disc Format, Label Verification, and Media Verification Test

This program performs in three operating modes that are dependent upon the System Disc Flag setting and option 19 (conditions program to operate in media verification mode). The three operating modes are as follows:

- Label verification, which verifies all the labels on the media.
- Format and label verification, which formats the media, verifies the labels, and deallocates defective tracks.
- Media verification, which formats the media, verifies the labels, then writes and reads every sector and deallocates defective tracks.

The default condition for the drive is System Disc Flag ON and option 19 reset, which causes the program to verify all the labels on the media. In this mode of operation, no deallocation information is written to the media.

3.2.1 Label Verification

System Disc Flag - ON
Option 19 - reset

This mode conditions the program to read all labels on the media and to report any errors other than label flag settings to the listed output (LO) file. Only the cylinder, head, and sector information in the first four bytes of the labels is checked to determine if the proper label is accessed.

The program writes the label flag settings that are detected as being set to anything other than a Good Track/Sector Label to the LO file. (Refer to Label Verification Messages.) The label verification mode is used once the media is formatted and verified. The defective tracks that are detected are not written to the media. New defective tracks that are detected are inserted into the Deallocation Track Table by using the ..,DT,.. directive. The ..,DT,.. directive allows for the insertion of defective tracks without verifying the media.

The test reads all labels on the track beginning with cylinder zero, head zero and ending with the last cylinder specified as parameter U and the last head specified as parameter K on the drive. The default track coordinate for this track is parameter G (maximum cylinder number) minus four, head zero, which is the first head on the drive, and sector zero. The deallocated track data is written listed to the LO file by using the ..,DBT,.. directive.

The following example shows how to condition the utility program for this operating mode:

Consider the following:

- The drive is a Model 8117 (32 MB) connected to device subaddress 6_{16}. The plug on the drive is set to three.
- The program is loaded with the name D1.
- Only the removable disc pack is to have labels verified.
- The sequence starts after the Assign procedure is performed.

```
DXP>RESET D1 OPT 19
DXP>D1 AU6,MOD8117, SDF='SYS(ON)',CMM
DXP>START D1
```

Note: The ..,CMM,.. directive conditions the utility program to verify only the removable disc pack. The default condition for the Cartridge Module (Model 8117) verifies the removable media.

3.2.2 Format and Label Verification

System Disc Flag - OFF
Option 19 - reset

This mode conditions the program to format the entire pack and verify all labels on the media in the same manner as decribed in the Label Verification mode.

Format and Label Verification mode is used initially to format the entire media. The media is formatted beginning with cylinder zero, head zero, and ends with the cylinder specified as parameter G minus one and the head specified in parameter K.

The following examples show how to condition the program for this operating mode:

Consider the following:

- The drive is a Model 8055 (80 MB) connected to device subaddress four. The plug on the drive is set to two.

- The program is loaded as D1.
- This sequence starts after the Assign procedure is performed.

Single Port

```
DXP>RESET D1 OPT 19
DXP>D1 AU4,MOD9342,'SDF='SYS(OFF)',SING
DXP>START D1
```

Dual Port

```
DXP>RESET D1 OPT 19
DXP>D1 AU4,MOD9342,SDF='SYS(OFF)',DUAL0
DXP>START D1
```

Note: The ..,DUAL0,.. and ..,SING,.. directives, allow for Single or Dual Port formatting. Sector formatting is changed by using the ..,FOR16,.. directive.

The test sequence for formatting the disc follows:

1. STZ,PT=L,1,PH=0,SSO,S,SPH,SPS,SEK='CC',FDK,..

   Performs a Rezero command, followed by selection of track parameter L, sets head and sector coordinates to zero, selects the head address in parameter H and sector address in parameter S, inserts a command chained Seek command in the I/O command list (IOCL) queue, inserts a format track command in the IOCL queue, and initiates a Start I/O instruction.

2. ..,PH+1,J2/K,..

   Increments the head address coordinate variable parameter by one and repeats the command chained seek/format disc/increment head sequence until all heads on track address coordinate parameter T are formatted.

3. ..,PT+1,J1/T>=U,:

   Increments the track address coordinate parameter T by one and repeats the seek/format disc/head increment/track increment/ until parameter T is greater than parameter U.

After formatting the entire media, the program reads all sector and track labels and reports any errors that are detected. All labels on the media should have a Good track format because the media have just been formatted.

The sequence follows:

1. Performs a Read All Sectors and Track Label on each track, and if no errors are detected, continues the same operation until the last track (parameter U) and last head (parameter K) are accessed. The program then writes the following summary to the listed output (LO) file:

   ```
   'taskname' ***** TRACK DEALLOCATION SUMMARY **CA:xx,DSA:y
   ```

xx specifies the two-character, hexadecimal current controller address.

y specifies the one-character, hexadecimal current controller subaddress.

2. If the described error conditions are detected, the following error messages are written to the listed output (LO) file:

```
CORRECTABLE DATA ECC
UNCORRECTABLE DATA ECC**ERROR**
DISC FORMAT ERROR**ERROR**
ECC IN SECTOR LABEL**ERROR**
ECC IN TRACK LABEL**ERROR**
NO SI RECEIVED**ERROR**
CPC OR INCR LENGTH**ERROR**
```

Refer to the Track Deallocation Messages for more information.

3. The program then deallocates two cylinders to be used as diagnostic cylinders, the last track on the media for hardware test equipment, and the cylinder that contains the deallocation data on the media. A pointer then is written in track label zero, head zero (bytes 12 through 15) pointing to the Deallocated Track Summary Table.

4. Both the Gould RPU Disc Processor Diagnostic and the designated operating system use the pointer in track label zero, head zero to determine where the deallocated tracks are on that particular media. The diagnostic uses the two deallocated cylinders to perform write/read tests if the system disc flag is ON. The operating system deallocates all the tracks or sectors in the Deallocation Track Table.

3.2.3 Media Verification

System Disc Flag - OFF
Option 19 - Set

This mode conditions the program to perform the following:

- Format the media as dual or single port.
- Verify all labels.
- Write data to all sectors on a track using a 5A5A5A5A pattern for all model number drives except Models 8136 and 8146 which use a C67BC67B pattern.
- Write sector labels to a full track.
- Read a full track of data using a 5A5A5A5A pattern for all model number drives except Models 8136 and 8146 which use a C67BC67B pattern.
- Write a full track of data using a Binary Progression.
- Write sector labels to a full track.
- Read a full track of data verifying the Binary Progression.

After formatting the media, the program performs the following steps:

Verification Procedure

Note: If a cache disc accelerator is being used to verify the media, all read data commands have a disabled automatic correction code command, which is command-chained to the read command. This allows all data errors to be detected without having the cache disc accelerator automatically correct data errors which are less than nine bits long.

1. Performs a read of all sector labels and track labels and compares the data input to a good track data prototype. If any errors are detected, the program jumps to Step 10.

2. Writes the data pattern 5A5A5A5A or C67BC67B to all sectors on the track using a single track write command. If any errors are detected, the program jumps to Step 10.

3. Writes all sector labels to the entire track as a label setting of good track. If any errors are detected, the program jumps to Step 10.

4. If a cache disc accelerator is being used to verify the media, the program continues with Step 4A. Otherwise, it reads the 5A5A5A5A or C67BC67B data pattern from all sectors on the track using a single read command. If any errors are detected, the program continues to Step 5.

4A. Performs a Disable ECC correction command-chained to a read command of a binary progression from all sectors on the track in a single Disable ECC and read operation. If any errors are detected, the program jumps to Step 10.

5. Writes a binary progression data pattern to all sectors on the track with a single write command. If any errors are detected, the program jumps to Step 10.

6. Writes all sector labels to the entire sector with a sector label setting of good track. If any errors are detected, the program jumps to Step 10.

7. If a cache disc accelerator is being used to verify the media, the program continues with Step 7A. Otherwise, it reads a binary progression from all sectors on the track in a single read operation. If any errors are detected, the program continues to Step 8.

7A. Performs a Disable ECC correction comand-chained to a read command of a binary progresssion from all sectors on the track in a single Disable ECC and read operation. If any errors are detected, the program jumps to Step 10.

8. The above procedures are executed until all tracks on the media are tested. After media verification, the program writes the list of deallocated tracks to the coordinates shown in the media descriptor Boundaries. The program then writes protect the track, which has the deallocated track data. This track always is reported as a write protected track if labels are verified.

Additional tracks may be added to the Deallocated Track Table by using the ..,DT,.. directive.

9. The program then writes a deallocation track and sector summary report of all the detected errors to the LO file. (Refer to Track Deallocation Messages). The coordinates of the deallocation track then are written to track label zero, head zero. After the track and sector summary is completed, the utility program exits the test.

Error Detection

If any errors other than "correctable" or "uncorrectable ECC error" are detected, proceed to Step 12. If a cache disc accelerator is being used to test the media, proceed to Step 11.

10. If a correctable data ECC error is reported, the program reads the data on the track using single sector read commands. A Read Error Correction command determines whether or not the error is correctable or uncorrectable. The program then either writes and reads a 5A5A5A5A or C67BC67B data pattern if the error resulted from Step 4 or uses a binary regression pattern if the error is detected in Step 7. In either case, an uncorrectable ECC error overrides any correctable ECC errors, and if the condition is detected, the sector is entered into the Deallocated Track Table.

    If a correctable ECC error in a sector is determined to be an uncorrectable ECC error by testing a different data pattern, the following message is written to the listed output (LO) file:

```
'taskname' CORRECTABLE ECC ERROR IS NOW UNCORRECTABLE ECC ERROR
```

Testing continues as described in Step 1.

11. If an ECC error is detected while verifying media using the cache disc accelerator, the error is entered into the Deallocted Track Table as an "uncorrectable ECC error". Testing continues as described in Step 1.

12. The following error conditions cause termination of track testing:

```
DISC FORMAT ERROR **ERROR**
ECC IN SECTOR LABEL **ERROR**
ECC IN TRACK LABEL **ERROR**
NO SI RECEIVED **ERROR**
CPC OR INCR LENGTH **ERROR**
```

    These errors are reported and the coordinates are entered into the Deallocated Track Table and testing continues as described in Step 1.

CHAPTER 4

ERROR REPORTING

4.1 Error Message Assignment

The Gould RPU Disc Processor Media Verification Program writes all error messages to the listed output (LO) file. The default LO file assignment is to the operator's console.

To assign the LO file to a device other than the operator's console, enter:

```
DXP>ASSIGN LO=ttaa [,dsa]
```

ttaa — specifies the four-character device mnemonic containing the two-character device type code and the two-character hexadecimal channel address.

dsa — specifies the optional two-character hexadecimal device subaddress (if the assignment is to a magnetic tape device).

4.2 Error Messages and Message Sequence

An error message sequence consists of:

1. The message separation banner

```
<<<<'taskname'----hh:mm:ss>>>>
```

taskname — specifies the one to eight character utility program name indicated in the load map.

hh:mm:ss — specifies the system elapsed execution time or the current time-of-day (if initialized) at the point of error.

2. The error message header

```
TST:ttt, CA:cc, DSA:uu, PC:ppppp

DRTV:xxxx
EXP COOR yyy--yy--yy STAR zzz--zz--zz
CMMD iiiiiiii IOCD dddddddd dddddddd
```

ttt — specifies the decimal test number of the erring test.

cc — specifies the hexadecimal Disc Processor channel address assigned.

uu — specifies the hexadecimal device subaddress assigned.

ppppp — specifies the program counter value within the erring test.

xxxx	specifies the last directive processed before the error was detected.
yyy--yy--yy	specifies the track, head, and sector disc coordinates as calculated by the utility program.
zzz--zz--zz	specifies the track, head, and sector disc coordinates as returned in sense word one. This value is the current contents of the Sector Target Address Register (STAR).
iiiiiiii	specifies the current instruction.
dddddddd	specifies the erring I/O command doubleword (IOCD).

3. The status error message

```
STATUS ERROR
EXP (UNIT)u (IOCD) bbbbbb (BITS) eee (COUNT) tttt
ECT (UNIT)u (IOCD) bbbbbb (BITS) eee (COUNT) tttt
```

EXP	indicates the expected status values.
ACT	indicates the actual status values.
u	specifies the hexadecimal device subaddress.
bbbbbb	specifies the address of the expected and actual data address for the I/O command list termination.
eee	specifies the status of the drive.
tttt	specifies the residual hexadecimal byte count from the last I/O command doubleword executed in the I/O command list queue.

3A. One of the following messages is written for each occurrence of a status bit in either the expected or actual status unless option 12 (inhibits status and sense data error message expansion) is set. If the message is preceded by a plus (+) sign, the status bit indicated is received but not expected; if preceded by a minus (-) sign, the status bit indicated is expected but not received.

Status Bit	Explanation
0	HIO AND STOP FUNCTION
1	PROGRAM CONTROLLED INTERRUPT
2	INCORRECT LENGTH
3	CHANNEL PROGRAM CHECK
4	CHANNEL DATA CHECK
5	CHANNEL CONTROL CHECK
6	INTERFACE CHECK
7	CHAINING CHECK
8	DEVICE BUSY
9	STATUS MODIFIER
10	N/A
11	N/A
12	CHANNEL END
13	DEVICE END
14	UNIT CHECK
15	UNIT EXCEPTION

3B. Sense Buffer Register (SBR, SNS Bytes 5, 6, 7)

One of the following messages is written for each occurrence of a sense word two data bit in either the expected or the actual status unless option 12 (inhibits error message expansion) is set. If the message is preceded by a plus (+) sign, the sense data bit indicated is received but not expected, and if it is preceded by a minus (-) sign, the sense data bit indicated is expected but not received.

Sense bits 0 through 7 of sense word two contain the current mode register setting.

Sense Bit	Explanation
8	COMMAND REJECT
9	OPERATOR INTERVENTION REQUESTED
10	N/A
11	EQUIPMENT CHECK
12	DATA CHECK
13	DATA OVERRUN OR UNDERRUN
14	DISC FORMAT ERROR
15	DEFECTIVE TRACK ENCOUNTERED
16	LAST TRACK FLAG ENCOUNTERED
17	ALTERNATE TRACK FLAG ENCOUNTERED
18	WRITE TO WRITE PROTECTED DISC
19	WRITE LOCK VIOLATION
20	MODE CHECK
21	INVALID MEMORY ADDRESS
22	RELEASE FAULT
23	CHAINING ERROR
24	LOST REVOLUTION
25	DISC ADDRESSING OR SEEK ERROR
26	BUFFER CHECK
27	ECC ERROR IN SECTOR LABEL
28	ECC ERROR IN DATA
29	ECC ERROR IN TRACK LABEL
30	RESERVED TRACK ACCESS ERROR
31	UNCORRECTABLE ECC ERROR

3C. Drive Related Status (SNS Bytes 12, 13)

0	SEEK END
1	UNIT SELECTED
2-7	ANGULAR POSITION COUNTER
8	FAULT
9	SEEK ERROR
10	ON CYLINDER
11	UNIT READY
12	WRITE PROTECTED
13	BUSY

4. The data error message

```
DATA ERROR(S) FR#: BFO: BFI:
fffff    dd    dd
  .      .     .
  .      .     .
  .      .     .
fffff    dd    dd
xxxxx OUT OF yyyyy
```

FR# indicates the frame number(s) in error.

BFO indicates the output buffer (expected) value(s).

BFI indicates the input buffer (actual) value(s).

fffff specifies the frame number of the first frame in error and each successive error up to five frames in error.

dd specifies the data value contained in the indicated buffer segment.

xxxxx specifies the total number of frames in error.

yyyyy specifies the total number of frames compared.

Note: The total number of frames compared is equal to the number of specified bytes in the initial transfer count (ITC). However, if the residual byte count (RBC) indicates that fewer than the requested number of transfers are performed, only the number of actually transferred frames is compared.

4.3 Status Error Report Exceptions

Status word checking is performed after all Start I/O (SIO) instructions are completed. If any discrepancies are detected between the expected status and the actual status, the status error(s) is reported. However, status words are not checked if any of the following error conditions are detected after a Start I/O (SIO) instruction is issued:

- Condition Code Errors
  If condition code errors occur, they are reported with no further status checking.

- No Interrupt or Device End Received Errors
  If an expected interrupt is not detected from the device, the no status interrupt message is reported with no further status checking.

- Status Address Word Errors
  If the status address is not correct, the program does not check the location for valid status information.

4.4 Condition Code Errors

Condition code errors are reported if they are detected after executing an Extended I/O instruction. Normal condition codes request accepted and queued, hexadecimal eight or status stored hexadecimal four are not reported. If a condition code error is detected, the following messages are written to the operator communications (OC) file:

If the current instruction being executed is not a Reset Channel instruction, the following message is written:

```
CC=0, UNASSIGNED
```

This condition code is valid only for the Reset Channel instruction.

If the channel timed out and remained busy without allowing for a request accepted or status stored condition, the following message is written:

```
CC=1, CHANNEL BUSY
```

If the channel/controller is not defined, the following message is written:

```
CC=2, UNDEFINED
```

If the subchannel timed out as busy and did not allow for a request accepted or status stored condition, the following message is written:

```
CC=3, SUBCHANNEL BUSY
```

If the Disc Processor channel is rejecting the current command as not being supported by the device, the following message is written:

```
CC=5, UNSUPPORTED
```

If an unassigned condition code is detected, the following message is written:

```
CC=x, UNASSIGNED
```

x equals hexadecimal 6,7,9,10,11,12,13,14,15.

These conditions codes are unassigned and should never be received.

If any of the above condition code errors are reported, no attempt is made to check the status information because there should be no status information received from the Disc Processor.

4.5 No Interrupt and Device End Messages

If an interrupt is expected and not received from the device or if no device end is reported after an interrupt is received from the device, one of the following messages is written to the OC file:

- If an Initialize Channel instruction failure occurs, the following message is written to the operator communications (OC) file:

  ```
  TERMINATED: CANNOT INITIALIZE CHANNEL
  ```

  This message indicates that the device cannot be initialized and that the program does not proceed until the device is initialized.

- If a Load Mode Register command failure occurs, the following message is written to the OC file:

  ```
  TST:LMR,CA:cc,DSA:ss,PC:aaaaa
  pppp,DRTV:LMR
  CMND:rrrrrrrr IOCD dddddddd dddddddd
  ```

  This message is accompanied with a status error and/or sense data errors. This message also may be reported if errors are detected during media changes on the Cartridge Module (Model 8117).

4.8 Additional Messages

Additional messages are written to the operator communications (OC) file if an unexpected error occurs:

If the program cannot correctly perform and initialize channel command to the device, the following message is written:

```
CANNOT INITIALIZE CHANNEL
```

This message usually is accompanied by a condition code error message or a no status interrupt message. No further testing can be performed until the channel is initialized.

If no interrupt is detected but an interrupt is expected during the execution of interrupt instruction testing, the following message is written:

```
NO SI
```

This message is exclusive to execution of Test 1.

If an interrupt is detected but an interrupt is not expected during the execution of interrupt instruction testing, the following message is written:

```
UNEXPECTED SI
```

This message is exclusive to execution of Test 1.

If a reset channel directive is performed an initialize channel command is not issued before attempting to perform another I/O directive, the following message is written:

```
CHANNEL MUST BE INITIALIZED
```

If during the execution of Test 2, no interrupt is detected after a Halt I/O instruction is executed, the following message is written:

```
IDLE STATE:  NO INTERRUPT AFTER HIO
```

If during the execution of Test 2, an interrupt is detected after a Stop I/O instruction is executed, the following message is written:

```
IDLE STATE:  INTERRUPT AFTER STP
```

If during the execution of Test 2, a Test I/O instruction does not indicate that the Test I/O instruction is accepted with no echo response, but rather gave the response as represented by an asterisk (*), the following message is written:

```
IDLE STATE:  EXPECTED CC 8
IDLE STATE:  ACTUAL CC *
```

* specifies the condition code received.

If after a Sense command is executed, the actual drive attributes are not what is expected from the drive, the following message is written:

```
ATTRIBUTE ERROR EXP=eeeeeeee, ACT=aaaaaaaa
```

eeeeeeee specifies the drive attributes expected from the disc drive.

aaaaaaaa specifies the actual attributes received from the disc drive.

This error occurs:

- If the drive is formatted as a dual port, the program is reset to single port, and an attempt is made to run the utility program in the single port mode without reformatting the disc.
- If a Cartridge Module Drive number is assigned to a 80 MB or 300 MB Disc Drive.

4.9 Directive Error Reporting

One of the following messages is written to the operator communications (OC) file if an assigned directive violates extended I/O protocol:

If more than 16 I/O command doublewords (IOCDs) are to be written into the I/O command list (IOCL) queue, the following message is written:

```
NUMBER OF DIRECTIVES PROCESSED WOULD HAVE CAUSED THE IOCD LIST TO OVERFLOW
```

The maximum number of IOCDs in the IOCL queue is 16.

If an attempt is made to change function during a data chain, the following message is written:

```
CANNOT DATA CHAIN TO A DIFFERENT TYPE OF I/O COMMAND
```

If the last I/O command doubleword (IOCD) entered in the I/O command list (IOCL) queue has its data chain flag set and no IOCD follows (incomplete chain), the following message is written:

```
DIRECTIVE STRING ENDED WITH DATA CHAIN SET
```

If the data-chained byte count exceeds the maximum size of the output buffer, the following message is written:

```
DATA CHAIN WOULD HAVE CAUSED OVERRUN OF THE OUTPUT BUFFER
```

If the program detects a Transfer-in-Channel (TIC) to a TIC command, the following message is written:

```
'TIC' TO 'TIC' IS INVALID
```

This sequence causes an IOCD validation error.

If a Transfer-in-Channel (TIC) command is the first IOCD entered in the IOCL queue, the following message is written:

```
'TIC' AS FIRST IOCD INVALID
```

If a Transfer-in-Channel (TIC) IOCD points forward to an IOCL queue slot into which no valid IOCD exits, the following message is written:

```
'TIC' POINTS TO EMPTY IOCL QUEUE SLOT
```

If a Reset Channel command is executed and no Initialize Channel command is entered, the following message is written:

```
CHANNEL MUST BE INITIALIZED
```

If an attempt is made to data chain to an illegal command, the following message is written:

```
CANNOT DATA CHAIN THIS COMMAND
```

If no entries are currently in the IOCL queue, the following message is written:

```
IOCL QUEUE EMPTY
```

If a dual port directive is issued to a subaddress which has Cartridge Module Model 8117 assigned to the device subaddress, the following message is written:

```
CANNOT DUAL PORT A CARTRIDGE MODULE DRIVE
```

If a Cartridge Module directive or directive test is attempted but the program is operating on a Disc Processor channel Model 9324, the following message is written:

```
PHASE 1 DOES NOT SUPPORT MODEL #8117
```

Cartridge Module Drive model 8117 is supported only by Disc Processor Model 8055.

If the media formatting is prematurely terminated by a program abort, Halt I/O, Stop I/O or Reset channel instruction, the following message is written:

SUBADDRESS * WAS TERMINATED DURING MEDIA FORMAT AT ????--??--??

* specifies the assigned subaddress.

? specifies the disc coordinates where termination took place.

If the ..,DBT,.. (Dump Bad Track) directive does not detect a label flag byte setting of Write Protect, the following message is written:

DEALLOCATED TRACK WAS NOT WRITE PROTECTED

This message may indicate that the information in the Deallocated Track Table is not valid.

If the first word of the Deallocated Track Table does not have an F0 in the first byte of data input from the media, the following message is written:

1ST WD OF BAD TRK HAS INCORRECT IDENTIFIER

This message indicates that the data on the sector is not valid and no list out of the deallocated tracks is reported. The contents of the Deallocated Track Table is displayed by using a Dump directive.

If a Cartridge Module directive (such as CCM, CMB, or CMF,) is entered via the operator's console and the program detected that a Cartridge model number assignment has not been made for the device subaddress reported, the following message is written:

SUBADDRESS * DOES NOT HAVE CARTRIDGE MODEL #8117

* specifies the one-character, hexadecimal currently assigned subaddress.

4.10 Label Verification Messages

During the execution of Test 1, track and sector label flags are checked for settings other than good track or sector. If any of these conditions are detected, the following messages are written to the listed output (LO) file:

Mnemonic	Represents
aaa	cylinder number
bb	head number
cc	starting sector number
ddd	defective or alternate cylinder
ee	defective or alternate head
ff	the total number of sectors starting at the value shown in cc

If bit 1 of label flag byte 1 is set, making this an alternate track which points to defective track ddd--ee, the following message is written:

```
'taskname' aaa--bb--cc ALTERNATE=DEFECTIVE TRACK ddd--ee ff SECTOR(S)
```

If bit 2 of label flag byte 1 is set, the following message is written:

```
'taskname' aaa--bb--cc SPARE TRACK ff SECTOR(S)
```

If bit 3 of label flag byte 1 is set, the following message is written:

```
'taskname' aaa--bb--cc RESERVED TRACK ff SECTOR(S)
```

If bit 4 of label flag byte 1 is set, making this a defective track which points to alternate track ddd-eee, the following message is written:

```
'taskname' aaa--bb--cc DEFECTIVE=ALTERNATE TRACK ddd--ee ff SECTOR(S)
```

If bit 5 of label flag type 1 is set, the following message is written:

```
'taskname' aaa--bb--cc LAST TRACK ff SECTOR(S)
```

If bit 0 of label flag byte 2 is set, the following message is written:

```
'taskname' aaa--bb--cc WRITE LOCK SET ff SECTOR(S)
```

If bit 1 of label flag byte 2 is set, the following message is written:

```
'taskname' aaa--bb--cc WRITE PROTECT SET ff SECTOR(S)
```

4.10.1 Description of Label Flag Byte One

Bit	Track	Description
0	Good Track	Most tracks are good tracks.
1	Alternate Track	This track is the alternate for a defective track.
2	Spare Track	This is an unused track which may be used later as an alternate track.
3	Reserved Track	This is a track which can only be read from or written to if the subchannel is in the reserve track mode. Also, there must be a jumper present on the disc controller to allow this access.
4	Defective Track	This track defined as unusable causes alternate track addressing to take place if the defective track is accessed.

5	Last Track	This bit may be set in combination with any of the orders. If a user specifies a byte count in an I/O command doubleword (IOCD) which extends past the end of this track, the I/O is terminated. In addition, if Defective Track is also set, then alternate track addressing does not take place.
6-7		Unimplemented.

4.10.2 Description of Label Flag Byte Two

Bit	Condition	Description
1	Write Protected	If this bit is set, it is not possible to write to the data portion of the sectors on this track.
0	Write Locked	If this bit is set, it is not possible to alter any of the labels on this track.

4.11 Track Deallocation Messages

If a track is deallocated, a message is written to the listed output (LO) file specifying the reason for deallocation. The standard track deallocated message format is as follows:

```
'taskname'  scyl-hh-ss  8raaaaaa      comment
            ecyl-hh-ss  9raaaaaa
```

taskname	specifies the one to eight character diagnostic name as indicated in the load map.
scyl-hh-ss	specifies the starting cylinder (scyl), head (hh) and sector (ss) of the deallocated track.
ecyl-hh-ss	specifies the ending cylinder (ecyl), head (hh) and sector (ss) of the deallocated track.
8	specifies the starting address. The absolute binary address (aaaaaa) in the value beginning with eight is the starting sector address of the deallocated track or sector.
9	specifies the ending address. The absolute binary address (aaaaaa) in the value beginning with nine is the ending sector address of the deallocated track or sector.
r	specifies the track deallocation reason code. The reason code (identified one through C) identifies which of the 12 reasons the track is deallocated.
aaaaaa	specifies the absolute binary address. This address is either the starting (preceded by an eight) or ending (preceded by a nine) binary sector address of the deallocated track in the format of the designated operating system.

comment — specifies the reason the track is deallocated:

1	ECC IN TRACK LABEL
2	ECC IN SECTOR LABEL
3	UNCORRECTABLE DATA ECC
4	CORRECTABLE DATA ECC
5	NO SI RECEIVED
6	CPC OR INCR LENGTH
7	DISC FORMAT ERROR
8	DIAGNOSTIC BOOT LOADER BOUNDARIES
9	DIAGNOSTIC CYLINDER BOUNDARIES
A	MEDIA DESCRIPTOR BOUNDARIES
B	ADDED FALSE ADDRESS MARK
C	HARDWARE DEVICE TEST TRACK

Reason Code 1 ECC IN TRACK LABEL

If bit 29 of sense word 2 is set after the utility attempts to adjust address marks by reformatting the track and attempts to read track labels, the following message is written:

```
'taskname' scyl-hh-ss  81aaaaaa   ECC IN TRACK LABEL**ERROR**
           ecyl-hh-ss  91aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 2 ECC IN SECTOR LABEL

If bit 27 of sense word 2 is set after the utility attempts to adjust address marks by reformatting the track and attempts to write or read data, the following message is written:

```
'taskname' scyl-hh-ss  82aaaaaa   ECC IN SECTOR LABEL**ERROR**
           ecyl-hh-ss  92aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 3 UNCORRECTABLE DATA ECC

If an error correction code (ECC) data error occurs, and after completing the Read ECC command, sense word 2 bit 31 is set, indicating that the data error is uncorrectable, the following message is written:

```
'taskname' scyl-hh-ss  83aaaaaa UNCORRECTABLE DATA ECC**ERROR**
           ecyl-hh-ss  93aaaaaa
```

The sector is considered to be a deallocated sector.

Reason Code 4 CORRECTABLE DATA ECC

If an error correction code (ECC) data error occurs, the following message is written:

```
'taskname' scyl-hh-ss  84aaaaaa  CORRECTABLE DATA ECC
           ecyl-hh-ss  94aaaaaa
```

However, if the error is corrected by using the Read ECC command and applying the displacement and correction mask to the data in error the sector is considered a potential deallocated sector.

Reason Code 5 NO STATUS INTERRUPT RECEIVED

If no interrupt is received while attempting to read track labels or data, the following message is written:

```
'taskname' scyl-hh-ss  85aaaaaa   NO SI RECEIVED
           ecyl-hh-ss  95aaaaaa
```

This message indicates that a false address mark may have been detected. This message also indicates that the track cannot be accessed by write or read commands and is, therefore, a deallocated track.

Reason Code 6 CHANNEL PROGRAM CHECK OR INCORRECT LENGTH

If after writing data to the media, the program attempts to read a track label that does not have a 30-byte count record size, the following message is written:

```
'taskname' scyl-hh-ss  86aaaaaa   CPC OR INCR LENGTH**ERROR**
           ecyl-hh-ss  96aaaaaa
```

This error also occurs if the track doesn't have the correct sector format, i.e., 16-sector format instead of 20-sector format.

Reason Code 7 DISC FORMAT ERROR

If sense word 2, bit 14 is set after the diagnostic attempts to adjust address marks by reformatting the track and attempts to read track/sector labels or data, the following message is written:

```
'taskname' scyl-hh-ss  87aaaaaa   DISC FORMAT ERROR**ERROR**
           ecyl-hh-ss  97aaaaaa
```

The track is considered to be a deallocated track.

Reason Code 8 DIAGNOSTIC BOOT LOAD BOUNDARIES

The following message is established for future use:

```
'taskname' scyl-hh-ss 88aaaaaa  DIAGNOSTIC BOOT LOAD BOUNDARIES
           ecyl-hh-ss 98aaaaaa
```

Reason Code 9 DIAGNOSTIC CYLINDER BOUNDARIES

If two cylinders that are defined during Media Verification and are used by the diagnostic program after executing the program with the System Disc Flag ON, the following message is written:

```
'taskname' scyl-hh-ss  89aaaaaa  DIAGNOSTIC CYLINDER BOUNDARIES
           ecyl-hh-ss  99aaaaaa
```

These two cylinders are also deallocated by the designated operating system and are devoted exclusively to the Gould RPU Disc Processor Diagnostic.

Reason Code A MEDIA DESCRIPTOR BOUNDARIES

The following message lists the deallocated track information within the boundaries of this cylinder:

```
'taskname' scyl-hh-ss  8Aaaaaaa   MEDIA DESCRIPTOR BOUNDARIES
          ecyl-hh-ss  9Aaaaaaa
```

The Gould RPU Media Verification Program establishes the deallocated tracks and verifies that the sectors which contain this information are error free.

Reason Code B ADDED FALSE ADDRESS MARK

If a track is deallocated by the ..,DT,.. directive, the following message is written:

```
'taskname' scyl-hh-ss  8Baaaaaa   ADDED FALSE ADDRESS MARK
          ecyl-hh-ss  9Baaaaaa
```

The ..,DT,.. directive allows the insertion of defective tracks to the Deallocated Track Table. The Gould RPU Disc Processor Diagnostic does not have the capability of adding false address marks by using the ..,DT,.. directive.

Reason Code C HARDWARE DEVICE TEST TRACK

If the last track on the media is deallocated to both the Gould RPU Disc Processor Diagnostic and the designated operating system, the following message is written:

```
'taskname' scyl-hh-ss  8Caaaaaa   HARDWARE DEVICE TEST TRACK
          ecyl-hh-ss  9Caaaaaa
```

This track is used exclusively by the hardware drive test equipment and should not be accessed by the system.

If a correctable ECC error in a sector is determined to be uncorrectable by testing a different data pattern, the following message is written:

```
'taskname' CORRECTABLE ECC ERROR IS NOW UNCORRECTABLE ECC ERROR
```

For each deallocated track or sector, one doubleword entry is written in the Deallocated Track Table. This entry contains the deallocated track or sector coordinates. If the Deallocated Track Table is full, the following message is written to the operator communications (OC) file:

```
DEALLOCATED TRACK TABLE FULL
```

CHAPTER 5

DIRECTIVE LANGUAGE

5.1 Introduction to the Directive Compiler

The Gould RPU Media Verification Program (RP.MVP.r) is structured around the capabilities of the Directive Language Compiler. The Directive Language Compiler (a component of the Diagnostic Executive Program), and its associated run-time package (a component of the diagnostic) use directive language to offer the following diagnostic capabilities:

- The ability to reduce a diagnostic test into its failing components for troubleshooting.
- The ability to create new directive test sequences.

The basic component of the directive language is the directive. For all Level Two diagnostics, there are two categories of directives -- standard and unique. The DEXP provides standard directives for all Level Two diagnostics, while each Level Two diagnostic contains unique directives to perform specific functions within the individual program.

A complete understanding of standard and unique directives is needed to use the directive language effectively.

5.2 DEXP Standard Directives

The Gould Diagnostic Executive Program contains nine standard directive categories. The following is a list of standard directives and their functions:

5.2.1 "A" Directives (Assign)

Three "A" directives assign channel, controller, and unit addresses for each controller.

..,A,..

> Uses the channel address and the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,AC,..
..,ACn,..
..,ACA,..
..,ACAn,..

> Uses the specified channel address n or, if n is not specified, the channel address currently in the device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

..,ADSA,..
..,ADSAn,..
..,AU,..
..,AUn,..

Uses the specified device subaddress n, or if n is not specified, the device subaddress currently in the diagnostic's device definition table (DDT) to create the Command Device (CD) and Test Device (TD) instructions.

5.2.2 "C" Directives (Cycle)

Four "C" directives cycle the output buffer (equivalent to a circular shift with end-to-end wraparound).

..,CL,..
..,CLn,..

Cycles the output buffer left by number n byte positions. If number n is not specified, the output buffer is cycled left by one byte position.

..,CR,..
..,CRn,..

Cycles the output buffer right by number n byte positions. If number n is not specified, the output buffer is cycled right by one byte position.

..,CLp,..

Cycles the output buffer left by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled left by one byte position.

..,CRp,..

Cycles the output buffer right by the value in p (any parameter A through Y). If the value in the specified parameter is zero, the output buffer is cycled right by one byte position.

5.2.3 "D" Directives (Dump)

Four "D" directives dump all or any specified part of the diagnostic's input buffer, output buffer or parameter set to the listed output (LO) file. If the LO file is assigned to a line printer, the byte count is rounded up to the next multiple of 32 bytes (eight words). Otherwise, the byte count is rounded up to the next multiple of 16 bytes (four words).

..,DI,..
..,DIn,..

Dumps number n or, if number n is not specified, parameter W bytes of the diagnostic's input buffer. If number n is not specified and parameter W equals zero, the entire input buffer is dumped.

..,DO,..
..,DOn,..

Dumps number n or, if number n is not specified, parameter W bytes of the diagnostic's output buffer. If number n is not specified and parameter W equals zero, the entire output buffer is dumped.

..,DP,..

Dumps the contents of all parameters (A through Z).

..,DPp,..

Dumps only the content of parameter p (any parameter A through Z).

5.2.4 "H" Directives (Hold)

Two "H" directives hold diagnostic execution. These directives establish delays between directives.

..,H,..
..,Hn,..

Holds diagnostic execution for the specified number of time units n or, if number n is not specified, for the number of time units in parameter X. If parameter X contains zero, the hold duration is one time unit (determined by the Real-Time Clock rate). If the Real-Time Clock rate is 120 Hz, one time unit equals 8.3 ms.

..,HS,..
..,HSn,..

Holds diagnostic execution for the specified number of seconds n or, if number n is not specified, for the number of seconds in parameter X. If parameter X contains zero, the hold duration is 1 second.

5.2.5 "J" Directives (Jump)

Five "J" directives transfer control in a directive string. The following are the relational operators to use with conditional jumps:

(op): - not
= equal to
> greater than
< less than
=> equal to or greater than
>= greater than or equal to
<= less than or equal to
=< equal to or less than

..,Jn,..

Jumps indefinitely to statement number n (0 <=n < =9).

..,JZ,..

Jumps indefinitely to the statement number in parameter Z.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /k,..

Jumps to statement number n or the statement number in parameter Z, k times (1<k<=32767).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p,..

Jumps to statement number n or the statement number in parameter Z the number of times defined by the contents of the specified parameter (A through Z).

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p(op)k,..

Jumps to statement number n or the statement number in parameter Z as long as the contents of p (any parameter A through Y) satisfies the stated relationship (op) with the constant k.

..,J $\left\{\begin{matrix} n \\ Z \end{matrix}\right\}$ /p_1(op)p_2,..

Jumps to statement number n or the statement number in parameter Z as long as the contents of p_1 (the first specified parameter A through Z) satisfies the stated relationship (op) with the contents of p_2 (the second specified parameter A through Y).

5.2.6 "L" Directives (Load)

Fifteen "L" directives load input or output buffers with a variety of data test patterns.

..,L=' '

Loads the output buffer with 'any literal value' and stores the literal string length in parameter W.

..,LA,..
..,LAn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the 64-character printable ASCII subset. If parameter W equals zero, the entire output buffer is loaded.

..,LIZ,..
..,LIZn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with all zeros. If parameter W equals zero, the entire input buffer is loaded.

..,LZ,..
..,LZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all zeros. If parameter W equals zero, the entire output buffer is loaded.

..,LO,..
..,LOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with all ones. If parameter W equals zero, the entire output buffer is loaded.

..,LFZ,..
..,LFZn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating zero (FE,FD,FB,...,BF,7F,FE,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LFO,..
..,LFOn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a floating one (01,02,04,...,40,80,01,...) pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LBK,..
..,LBKn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with ASCII blanks (20,20,20,...,20,20,20,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBP,..
..,LBPn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary progression (00,01,02,...,FE,FF,00,...). If parameter W equals zero, the entire output buffer is loaded.

..,LBR,..
..,LBRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with a binary regression (FF,FE,FD,...,01,00,FF,...). If parameter W equals zero, the entire output bufffer is loaded.

..,LIR,..
..,LIRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the initial random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LLR,..
..,LLRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the last random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LNR,..
..,LNRn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the next random number pattern. If parameter W equals zero, the entire output buffer is loaded.

..,LOI,..
..,LOIn,..

Loads number n or, if number n is not specified, parameter W bytes of the output buffer with the contents of the input buffer. If parameter W equals zero, the entire output buffer is loaded.

..,LIO,..
..,LIOn,..

Loads number n or, if number n is not specified, parameter W bytes of the input buffer with the contents of the output buffer. If parameter W equals zero, the entire input buffer is loaded.

5.2.7 "P" Directives (Parameter)

Five "P" directives manipulate entries in the diagnostic's parameter table. The following are arithmetic and logical operators to use with "P" directives:

(op):	=	replacement
	+	addition
	-	subtraction
	*	multiplication
	/	division
	.A.	logical AND
	.O.	logical OR
	.E.	logical EXCLUSIVE OR
	.N.	two's complement
	.C.	one's complement
	>	circular shift right
	<	circular shift left

..,Pp(op)k,..

The contents of the specified parameter (A through Z) is fetched then the logical OR arithmetic operation is performed. The result is stored into the specified parameter (the range for k is zero through 32,767).

..,Pp_1(op)p_2,..

The contents of the specified parameters p_1 and p_2 are fetched, then the logical OR arithmetic operation is performed. The result is stored in the specified parameter p_1.

..,PD(op) $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$,..

The contents of parameter D (and the second parameter if also specified) are fetched, then the logical OR arithmetic operation is performed. The result then is stored in parameter D and also into the entire output buffer.

..,PI(op) $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$,..

The contents of parameter I and either the second parameter (Pp) (if specified) or k are fetched then the logical OR arithmetic operation is performed. The result is stored into parameter I and also into word one of the output buffer.

..,PZ= $\left\{ \begin{matrix} k \\ Pp \end{matrix} \right\}$,..

The contents of Pp (P specifies any parameter A through Y) or k, is stored into parameter Z. The value of parameter Z represents a statement number to which the ..,JZ,.. directive transfers control. The only valid operator for this directive is = (replacement). The contents of the specified parameter must be between zero and nine.

5.2.8 "S" Directive (Stop)

The "S" directive creates convenient breaks in a directive string for performing necessary functions.

..,STPn,..

Stops diagnostic execution and issues the following prompt to the operator communications (OC) file:

```
'taskname' ,STPn, DIRECTIVE ; [C,H,T] ?
```

5.2.9 "T" Directive (Time)

The "T" directive writes the time-of-day and an informative message (if specified in the directive) to the listed output (LO) file.

..,TIME,..
,..TIME (message),..

5.3 Gould RPU Media Verification Unique Directives

The Gould RPU Media Verification Program contains unique directives to perform specific functions within the diagnostic. Many unique directives, however, cross diagnostic boundaries and are used in different diagnostics.

The following is a list of the Gould RPU Media Verification Program unique directives:

5.3.1 "A" Directives

..,ADP,..

Assigns the correct set of disc drive parameters for each specified drive in an ..,AU,.. (assign unit) directive string. These parameters consist of media size and beginning and ending addresses of media verification.

Note: This directive should always immediately follow the ..,AU,.. directive in a string. This insures that the correct parameters are selected each time a different model drive is assigned.

..,AMCn,..

Sets the first address mark n bytes from the index pulse in the track format address mark data used by the Format Disc command. The directive should be used prior to the Format Track directive ..,FT=,.. to adjust address marks after a track has been found to have a media flaw. The address mark adjustment remains in the track format address mark data until the ..,AMO,.. directive is used to reset the first address mark to the original data value of 21 bytes.

Consider the following:

- The program has the name D1.
- The drive is connected to subaddress 6.
- Cylinder 100, head 2 has been found to have a media flaw.
- The address mark adjustment is 70.

DXP > D1 AU6,AMC70,FT='THC(100,2)'

This command should be followed by resetting the address marks back to the original data:

DXP > D1 AU6,AMO

..,AMO,..

Sets the first address mark in the track format address mark data to 21 bytes from the index pulse (the default condition).

..,ANG,..
..,ANG='arg_1,arg_2,...,arg_n',..

Inserts a Read Angular Position (ANG) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed. For a description of the optional subfields, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,AUn,..

Assigns the device subaddress specified by n as the device subaddress of the disc drive to test and aligns the assigned device subaddress with the correct number of parameters for the specified drive. This directive is not needed if only one disc drive is configured.

Note: This directive should be immediately followed by the ..,ADP,.. (assign drive parameter) directive to insure that the correct parameter set is selected for the drive being assigned.

5.3.2 "B" Directives

..,BFE,..

Relocates the utility program's input buffer to the starting address of the output buffer. This directive is used to input data from one drive and have the data positioned for output to another drive. The ..,BFI=PGM,.. and ..,BFO=PGM,.. directives reset the starting addresses of the input and output buffers, respectively.

..,BFI='address',..

Relocates the utility program's input buffer to the specified address. A difference of 3000 should be allowed between the input and output buffers to insure that maximum byte count transfers do not cause buffer overflows.

..,BFI='PGM',..

Relocates the utility program's input buffer to the default location within the utility program.

..,BFO='address',..

Relocates the utility program's output buffer to the specified address.

..,BFO='PGM',..

Relocates the utility program's output buffer to the default location within the utility program.

5.3.3 "C" Directives

..,CA=,..

Converts from cylinder, head, and sector disc coordinates to absolute binary sector address.

..,CCM,..

Sets parameter Y to the number of fixed heads in the Drive Attribute Table.

..,CDA,..

Sets the cache disc accelerator (CDA) flag in parameter C.

..,CDP,..

Sets parameter B to the value of parameter K if the drive is not dual ported or if the System Disc Flag is OFF. If the drive is dual ported or the System Disc Flag is ON, parameter B has the value of parameter K minus 1.

..,CEC,..

Fetches the error correction data word from a Read Error Correction Code (ECC) command. The byte displacement is calculated from the bit displacement of the ECC word and the mask is Exclusive Orred into the first default parameter V bytes of the input buffer.

..,CMF,..

Conditions a flag which indicates to the utility program that disc drive access may be made only to the captive (fixed) media on the cartridge module drive. Once this directive is issued, the utility program tests and accesses only the captive media portion of the drive. The disc coordinates accessed by the utility program are dependent upon the System Disc Flag setting for this subaddress. If more than one drive is connected to the Disc Processor channel, this directive is preceded by the ..,AU,.. directive.

Example:

```
DXP > D1 AU4,MOD8117,CMF
```

..,CMM,..

Conditions a flag which indicates to the utility program that disc drive access may be made only to the removable disc pack on the cartridge module drive. This directive causes the utility program to test and access only the removable media on the drive. The disc coordinates accessed by the utility program are dependent upon the System Disc Flag setting for this subaddress. If more than one drive is connected to the Disc Processor channel, this directive is preceded by the ..,AU,.. directive.

Example:

```
DXP > D1 AU6,MOD8117,CMM
```

..,CNO,..

Clears parameter Y. Fetches the fixed head number if it is a fixed head cartridge module and stores it in parameter Y.

..,COM,..

Compares the contents of the input buffer with the contents of the output buffer and reports any errors detected.

..,CONn,..

Conditions a flag which indicates to the utility program which type of controller is being used by the drive. The utility program automatically defaults to controller Model 8055. This directive allows the utility program to function with either controller Model 8055 or 8060. The functionality of the program does not change for Model 8055 or 8060.

n represents:

8055	Phase 2 Controller model number (2 board)
8060	Phase 2 Controller model number (1 board)

..,CSF,..

Selects fixed head from the attributes and sets the flag for fixed head disc access only.

..,CSM,..

Calculates the maximun allowable heads.

..,CVF,..

Uses the value in parameter C to calculate the next head coordinate being used by the fixed portion of the cartridge module. This directive also establishes the current values for parameters F and K by using the number of fixed heads obtained from the drive attribute register.

..,CVM,..

Uses the value in parameter B to calculate the next head coordinate being used by the removable portion of the cartridge module. This directive also establishes the current values for parameters F and K by using the number of moving heads obtained from the drive attribute register.

..,CVX,..

Sets the end of test bit in the drive parameter table.

..,CWC,..

Returns the worst case test pattern established by the ..,CWO,.. directive.

..,CWN,..

Establishes a worst case test pattern which is the complement of the pattern established by the ..,CWO,.. directive.

..,CWR,..

Uses a worst case test pattern established by the ..,CWO,. directive and rotates it one bit.

..,CWO,..

Sets a 5A5A5A5A worst case test pattern for all drives except model numbers 8136 and 8146 which use a C67BC67B worst case test pattern.

5.3.4 "D" Directives

..,DAL0,..
..,DUAL0..,

Dual Port Using Entire Disc

Conditions parameters F, G, K, L, O, Q, and U to the default values indicated by the model number selected. The utility program then performs Reset Channel and Enable Channel interrupt instructions followed by an Initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the dual port mode. Refer to Disc Drive Test Parameters.

If more than one drive is connected to the Disc Processor channel, this directive should be preceded by the ..,AU,.. directive. This directive is used to format the media prior to dual port testing. This directive always should be preceded by the System Disc Flag directive to insure that the disc drive test parameters are set properly.

..,DAT,..

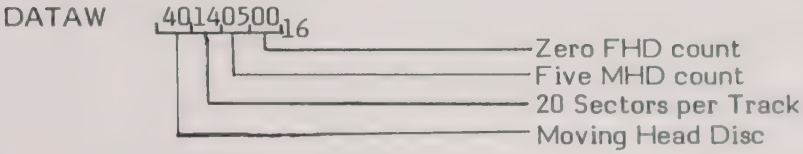
Writes the contents of all Drive Attribute Registers to the listed output (LO) file.

Drive Attribute Register layout is as follows:

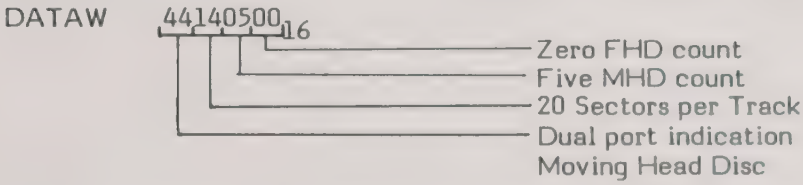
Bits	Meaning
0	10 = FHD (Fixed Head Disc)
1	01 = MHD (Moving Head Disc)
	11 = MHD with FHD option
	00 = Reserved
2	1 = Cartridge Module Drive
3	Reserved
4	1 = Drive Not Present
5	1 = Drive is Dual Ported
6-7	Reserved for future expansion, must be zero

Sector Count	specifies the number of sectors per track (14_{16}).
MHD Count	indicates the number of heads on the MHD or the number of heads for the removable disc pack portion of the cartridge module drive.
FHD Count	indicates the number of heads for the captive disc portion of the cartridge module drive.

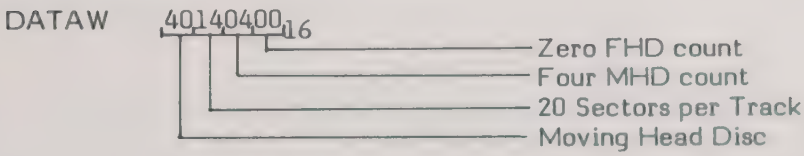
The Drive Attribute Register for a Model 9340 (40 megabyte) single port, moving head disc is generated as follows:



The Drive Attribute Register for a Model 9342 (80 megabyte) dual port, moving head disc is generated as follows:

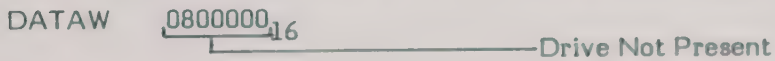


The Drive Attribute Register for a Model 8104 (5 megabyte) single port, fixed media disc is generated as follows:



The Drive Attribute Register for a Model 8117 (32 megabyte) cartridge module drive is generated as follows:

The Drive Attribute Register for a drive that is not present is generated as follows:



..,DBT,..

Dumps the Deallocated Track Table (bad tracks) by reading the deallocated track pointer from track label zero, then seeks to that track and writes the contents of the table to the listed output (LO) file. For the output messages format, refer to the Track Deallocation Messages.

Note: This directive should be preceded by both the ..,AU,.. (assign unit) and ..,ADP,.. (assign drive parameters) directives.

..,DDP,..

Dumps all the current values of the drive parameter table (parameter A through Z).

..,DEC,..
..,DEC='arg_1,arg_2,...,arg_n',...

Inserts address of disabled Error Correction Code (ECC) I/O Command Doubleword (IOCD) in the I/O Command List (IOCL) queue. This command is used with the cache disc accelerator to disable the automatic correction of ECC errors. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

..,DMP_n,..

Dumps the current data value sector to the listed output (LO) file and continues dumping sectors until number n sectors are dumped, or the end of the media is reached. If no number is specified, the entire disc from the current location to end of media is dumped.

..,DNEW,...

This directive converts bad track information from the old format (i.e., media) formatted by Revision C-D of the Gould RPU Disc Processor Diagnostic to the new deallocation track format which is used by the designated operating system and by the Gould RPU Disc Processor Diagnostic. The System Disc Flag must be turned OFF by using the ..,SDF,.. directive.

This directive reads the information in the Bad Track Table and automatically converts the information to the new deallocation track format. Default parameters for the diagnostic tracks, deallocation track and hardware test track also are entered into the Deallocation Track Table and the information is written to the cylinder designated as the Media Descriptor Boundaries. Track label zero, head zero is written to the media with a pointer to the deallocation track coordinates.

Information in the deallocation track then is deleted (if needed) by using the ..,XDT,.. directive or additional entries are entered by using the ..,DT,.. directive.

The procedure for using the ..,DNEW,.. directive is as follows:

```
SDF='SYS (OFF)',DNEW
```

Note: If deallocation track information is inadvertently destroyed, the ..,DNEW,.. directive is used to establish the media descriptor boundaries, diagnostic cylinder boundaries and the hardware device test track. Additional deallocated tracks are entered into the Deallocation Track Table by using the ..,DT,.. directive.

If the system disc flag is not set to OFF, the following message is written to the listed output (LO) file:

PLEASE SET SYSTEM DISC FLAG OFF OTHERWISE DEALLOCATION TABLE WILL NOT BE WRITTEN TO THE MEDIA

..,DQ,..

Dumps the contents of the I/O command list (IOCL) queue to the LO file.

..,DRC,..

Writes the following reason code for the ..,DT,.. directive to the LO file:

TO ADD ENTRIES TO A DEALLOCATION TABLE, USE THE DT DIRECTIVE

STEP 1. SDF='SYS(OFF)'
STEP 2. DT='THS(X,Y,Z),REAS(Q)

WHERE X=TRACK,Y=HEAD,Z=SECTOR,Q=REASON CODE

Reason Codes 9, A, and C are deallocated automatically if the ..,DNEW,.. directive is used.

REASON CODE	TYPE OF DEALLOCATION
1	TRACK LABEL ECC ERROR
2	SECTOR LABEL ECC ERROR
3	UNCORRECTABLE DATA ECC ERROR
4	CORRECTABLE DATA EXCC ERROR
5	NO SI RECEIVED
6	CPC OR INCORRECT LENGTH ERROR
7	DISC FORMAT ERROR
8	DIAG BOOT LOAD BOUNDARIES
9	DIAG CYLINDER BOUNDARIES
A	MEDIA DEALLOCATION CYLINDER
B	ADDED FALSE ADDRESS MARK
C	HARDWARE TEST TRACK

USE THE DRC DIRECTIVE TO LIST THE REASON CODES FOR DT DIRECTIVE USAGE

..,DSD,..

Dumps the sense data received from the last executed Sense command to the LO file.

..,DST,..

Dumps the channel/device status received from the last executed Start I/O (SIO) command to the LO file.

..,DT='THS(a,b,c),REAS(d)',..

This directive is used to insert additional entries into the Deallocation Track Table. The System Disc Flag must be OFF, otherwise the data is not written to the media.

This directive reads track label zero, head zero and uses the binary address to establish the coordinate of the Deallocation Track Table within the Media Descriptor Boundaries. The program then performs a Seek and Read command to input the Deallocation Track Table. The table then is updated with the specified values and the new Deallocation Track Table is written within the Media Descriptor Boundaries.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.
c specifies the sector.
d specifies the code for any of the reasons listed as follows:

1	ECC IN TRACK LABEL
2	ECC IN SECTOR LABEL
3	UNCORRECTABLE DATA ECC
4	CORRECTABLE DATA ECC
5	NO SI RECEIVED
6	CPC OR INCR LENGTH
7	DISC FORMAT ERROR
8	DIAGNOSTIC BOOT LOADER BOUNDARIES
9	DIAGNOSTIC CYLINDER BOUNDARIES
A	MEDIA DESCRIPTOR BOUNDARIES
B	ADDED FALSE ADDRESS MARK
C	HARDWARE DEVICE TEST TRACK

This directive is used to add to the Deallocation Track Table tracks which have False Address Marks not detected by the Gould RPU Media Verification Program. This insures that potential False Address Marks are entered into the Deallocation Track Table and their removal from the system is made by the designated operating system. Reason code B is used with this directive to add false address marks to the Deallocation Track Table.

The procedure for using the deallocate track directive is as follows:

SDS='SYS(OFF)',DT='THS(a,b,c),REAS(d)'

If the system disc flag is not set to OFF, the deallocate directive is terminated and the following message is written to the LO file:

PLEASE SET SYSTEM DISC FLAG OFF OTHERWISE DEALLOCATION TABLE WILL NOT BE WRITTEN TO THE MEDIA.

5.3.5 "F" Directives

..,FAM='THS(TRK(xxx),HD(yy),(zz)SEC)',..

This directive is used to insert additional entries, specifying false address marks in the deallocation track table.

xxx specifies the track number.
yy specifies the head number.
zz specifies the sector number.

..,FDK,..
..,FDK='arg_1,arg_2,...,arg_n',..

Inserts a Format Disc (FDK) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,FORn,..

Selects either a 16- or 20-sector track format for the assigned device. The permissible values for n are 20 (for a 20-sector track format) and 16 (for a 16-sector track format). If this directive is used, the following steps are performed by the utility program:

1. Sets up the drive attributes to the 16- or 20-sector data values specified.
2. Executes a Reset Channel instruction.
3. Executes an Enable Channel interrupt instruction.
4. Initializes the channel to the new device attribute:

Sector Format Type	Bytes/Sector
n=16	1024
20	768

..,FT='THC(a,b)',..

Automatically seeks to track address a and head address b, and reformats the specified track. This directive is used to format a defective track.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

5.3.6 "H" Directive

..,HELP,..

Writes pertinent information to the LO file.

..,HIO,..

Issues a Command Device (CD) Halt I/O instruction.

5.3.7 "I" Directives

..,IHA,..
..,IHA='arg_1,arg_2,...,arg_n',..

Inserts an Increment Head Address (IHA) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,INCH,..
..,INCH= ' arg_1,arg_2,...,arg_n',..

Inserts an Initialize Channel (INCH) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

5.3.8 "L" Directives

..,LAD,..
..,LADn,..

Loads n number of bytes, or if n is not specified, parameter W bytes of the diagnostic's output buffer with the track, head, and sector address coordinate values. If parameter W equals zero, the entire buffer is loaded.

..,LID,..
..,LIDn,..

Loads the current identifier track, head, and sector address coordinates, TRKHDSEC, into the first word of the diagnostic's output buffer, then if n is specified, calculates and inserts n number of successive identifier values into the output buffer. Words two through 192 of the output buffer remain unchanged.

..,LLS='arg_1,arg_2,...,arg_n',..

Creates an output buffer of sector labels modified by the specified arguments. For a description of the optional specifications refer to Track and Sector Flag Byte Arguments.

..,LLT,..
..,LLT='arg_1,arg_2,...,arg_n',..

Creates an output buffer of one track label modified by the specified arguments. The default condition for this directive is to set the track label to Good in the Track Flag byte. For a description of the optional specifications, refer to Track and Sector Flag Byte Arguments.

Rules:

- The sequence for preparing the output buffers is sector labels first, followed by the track label.
- The utility program uses the current contents of the utility program disc coordinate to build sequential sector addresses.
- The ..,LLS,.. and ..,LLT,.. directives are used to prepare an output buffer prior to the read track or read sector labels to verify that the labels are correct if the utility program performs a data compare.

..,LMR,..
..,LMR=',arg_1,arg_2,...,arg_n',..

Inserts a Load Mode Register (LMR) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default condition of this directive is to clear the mode register to zero. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to Load Mode Register Arguments.

..,LOOP,..

Sets the string address of this directive to the error return address. This allows keyboard Loop on Errors from this point in the directive string if option 3 is set and an error is detected.

..,LPL,..
..,LPL='arg_1,arg_2,...,arg_n',..

Inserts a Lock Protected Label (LPL) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,LRA,..

Uses parameter B as an index into the output buffer and selects a random number. If not dual ported or if a cartridge module and the system disc flag is OFF, it uses parameter O to calculate the track address. If dual ported or not a cartridge module parameter L is used as the starting parameter to calculate the track address. The track head sector coordinates are inserted into word 1 of the output buffer.

5.3.9 "M" Directive

..,MODn,..

Initializes the disc drive and test parameters of the current run-time subaddress to the default condition of the system disc flag ON and single port mode of operation. If more than one type of disc drive is to be tested, the run-time assign directive ..,AU,.. with the prescribed subaddress should precede the model number directive. This enables the utility program to set up the proper disc drive parameters for up to eight drives (device subaddresses) with any mixture of model numbers. The valid model numbers are listed as follows:

MOD8104	Fixed Head (5 megabyte) Disc Drive
MOD8117	Cartridge Module (32 megabyte) Disc Drive
MOD8122	Winchester (80 megabyte) Disc Drive
MOD8127	Winchester (160 megabyte) Disc Drive
MOD8136	Nine Inch Cartridge (80 megabyte) Disc Drive
MOD8146	Nine Inch Winchester (160 megabyte) Disc Drive
MOD8155	Captive Media (675 megabyte) Disc Drive
MOD8420	Eight Inch Winchester (80 megabyte) Disc Drive
MOD8424	Eight Inch Winchester (167 megabyte) Disc Drive
MOD8858	Nine Inch Winchester (340 megabyte) Disc Drive
MOD9342	Moving Head (80 megabyte) Disc Drive
MOD9344	Moving Head (300 megabyte) Disc Drive

Other directives that may follow the model number directive to set appropriate parameters for specific testing are the ..,CMF,.. and ..,CMM,.. directives for the Cartridge Module and the ..,CON,.. directive, which allows the utility program to test Disc Processor Channel/Controller Model 8055.

Note: The System Disc Flag is always ON if this directive is processed.

Consider the following:

- An 80 MB disc drive is connected to device subaddress two.
- Device subaddress two is to be dual port.
- Device subaddress two is to have the System Disc Flag OFF.
- A Cartridge Module Drive is connected to subaddress four.
- Device subaddress four is to have the System Disc Flag OFF.
- Only the removable disc pack on the Cartridge Module is to be tested.
- The program was loaded under the name D1.
- The Disc Processor Channel address is 08.

The procedure is as follows:

1. ASSIGN D1,DM08,2,4

2. Return control to the Diagnostic Executive Program (DEXP) by entering @@A on the IOP console.

3. DXP > D1,AU2,MOD9342,SDF='SYS(OFF)', DUAL0(cr)

4. The WHAT NEXT ?? prompt is issued to the operator communications (OC) file.

5. Return control to the DEXP by entering @@A on the IOP console.

6. DXP > D1,AU4,MOD8117,SDF='SYS(OFF)',CMM(cr)

7. Start the test.

5.3.10 "N" Directive

..,NOP,..
..,NOP='arg_1,arg_2,...,arg_n',..

Inserts a No Operation (NOP) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

5.3.11 "R" Directives

..,R,..
..,Rn,..

Inserts a Read I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The utility program reads the number of bytes specified by n, or if n is not specified, uses the value specified in parameter W as the byte count.

All Read commands use the current track, head, and sector address coordinates contained in the sector target address register (STAR).

..,RCH,..

Issues a Command Device (CD) Reset Channel instruction.

..,RCT,..

Issues a Command Device (CD) Reset Controller instruction.

..,REC,..
..,REC='arg_1,arg_2,...,arg_n',..

Inserts a Read ECC I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

..,REL,..
..,REL='arg_1,arg_2,...,arg_n',..

Inserts a Release (REL) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,REV,..
..,REV='arg_1,arg_2,...,arg_n',..

Inserts a Reserve (REV) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,RLS,..
..,RLS='arg_1,arg_2,...,arg_n',..

Inserts a Read Sector Label (RLS) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,RLT,..
..,RLT='arg_1,arg_2,...,arg_n',..

Inserts a Read Track Label (RLT) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,RRn,..
..,RR='arg_1,arg_2,...,arg_n',..

Inserts a Read I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The utility program reads the number of bytes specified by n, or if n is not specified, parameter W bytes. If parameter W is less than four bytes, the utility program defaults to four bytes. If command chain (CC) or data chain (DC), is specified, the appropriate bit in IOCD word 2 is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,RRM,..
..,RRM='arg_1,arg_2,...,arg_n',..

Inserts a Reset Reserved Track Mode I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

5.3.12 "S" Directives

..,SDF='SYS$\begin{Bmatrix}(ON)\\(OFF)\end{Bmatrix}$',..

Setting the System Disc Flag to OFF enables the utility program to access all tracks and heads on the selected model number drive. This directive should always precede the ..,DUALO,.. (dual port) directive or the ..,SING,.. (single port) directive. This directive enables the utility program to write/read the entire media. The starting and ending disc coordinates are dependent on the System Disc Flag setting. (Refer to Disc Drive Test Parameters.) Each drive has its own System Disc Flag setting, which allows selective testing of the media. Setting the System Disc Flag to OFF is used with discretion under the following conditions:

- The media must be formatted.
- The media must be formatted and verified.

If more than one type of drive is being tested, the System Disc Flag directive should be preceded by the run-time assign directive ..,AU,.. with the correct device subaddress for that drive.

Example:

```
DXP > D1 AU4, MOD9342, SDF='SYS (OFF)'
```

Setting the System Disc Flag OFF enables the utility program to format the entire media by the model number of the drive. The default System Disc Flag setting is ON.

..,SDP,..

Sets the dual port drive attributes.

..,SEK,..
..,SEK='arg_1,arg_2,...,arg_n',..
..,SK='arg_1,arg_2,...,arg_n',..

Inserts a Seek I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The track coordinate is obtained from the current contents of TRKHSEC and placed into the disc coordinate table with the data address of the Seek command pointing to the coordinate in the table. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,SIN,..

Issues a Command Device (CD) Start I/O instruction without waiting for an interrupt.

..,SING,..
..,SNG,..

Conditions parameters F, G, K, L, O, and U to the original values indicated by the model number selected. The utility program then performs a Reset Channel instruction and an Enable Channel Interrupt instruction, followed by an Initialize Channel command. The Initialize Channel command conditions the channel/device to operate in the single port mode (bit 5 reset in the drive attribute table). Refer to Disc Drive Test Parameters for system disc flag settings and drive test parameters.

..,SIO,..

Issues a Command Device (CD) Start I/O instruction.

..,SH(n),..

Sets the head address coordinate (n) into parameter H, which is used for subsequent Seek commands (where 0=<(n)<=K).

..,SNS,..
..,SNS='arg_1,arg_2,...,arg_n',..

Inserts a Sense I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,SPH,..

Sets the head address coordinate contained in parameter H into TRKHDSEC for subsequent Seek commands (where 0=<H<=K).

..,SPO,..
..,SPO='arg_1,arg_2,...,arg_n',..

Inserts a Priority Override I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to the IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,SPS,..

Sets the sector address coordinate in parameter S into TRKHDSEC (where 0=<S<=F).

..,SRM,..
..,SRM='arg_1,arg_2,...,arg_n',..

Inserts a Select Reserved Track Mode I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to the IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,SSn,..

Sets the head address coordinate n into parameter S, which is used for subsequent seek commands (where 0=<n<=K).

..,SSP,..

Sets the single port drive attributes.

..,STn,..

Sets the track coordinate n to a dummy parameter, which is used to test sector and track label defective/alternate, write/read commands (where 0 <n<U).

..,STZ,..
..,STZ='arg_1,arg_2,...,arg_n',..

Inserts a Rezero I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

5.3.13 "T" Directives

..,TDA='DEF (a,b), ALT (x,y)',..

a specifies the defective track.
b specifies the defective head.
x specifies the alternate track.
y specifies the alternate head.

Automatically seeks to track address a and head address b, and conditions the track and sector flag byte labels to indicate that the track is a defective track (bit 4 set). The alternate track address x and the alternate head address y are inserted into the appropriate fields in all the labels. The program then seeks to track x and head y and conditions the track and sector flag bytes labels to indicate that the track is an alternate track (bit 1 set). In addition, the defective track a and the defective head b are inserted into the appropriate fields in all the labels.

This directive always terminates the directive string.

..,TDF,..

Writes a message specifying the current setting of the System Disc Flag to the listed output (LO) file as follows:

SYSTEM DISC FLAG CA:xx, DSA:y(ON,OFF)

xx specifies the two-character, hexadecimal current controller address.
y specifies the one-character, hexadecimal current controller subaddress.

..,TES,..
..,TES='arg_1,arg_2,...,arg_n',..

Inserts a test sector target address register (STAR) I/O command doubleword (IOCD) into the I/O command list (IOCL) queue. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,TGOOD='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to Good track (bit 0 set). Alternate and defective track and head fields are cleared to zero.

a specifies the track coordinate.
b specifies the head coordinate.

..,TICn,..

Inserts a Transfer-in-Channel (TIC) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If the TIC is executed, control is transferred to the IOCD in the IOCL queue slot number specified by n. A TIC is an implied command chain (CC) operation, as the flag byte of the previous IOCD is passed through the TIC. Therefore, a TIC may never be the first IOCD in the IOCL queue.

..,TIO,..

Issues a Command Device (CD) Test I/O instruction.

..,TLAST='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to last track (bit 5 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

..,TLOK='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to the write lock (bit 0 of Label Flag byte 2) mode.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

..,TPRO='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to the write protect mode.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

..,TREV='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to reserved track (bit 3 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

..,TSP,..
..,TSP='arg_1,arg_2,..,arg_n',..

Inserts a Test Sector Target Address Register (STAR) I/O command doubleword (IOCD) into the I/O command list (IOCL) queue and sets appropriate flags. If command chain (CC) is specified, IOCD word 2, bit 1 is set and the next directive is processed.

The legal argument for this directive is: CC

..,TSPAR='THC (a,b)',..

Automatically seeks to track address a and head address b, and sets the flag byte labels in the track and sector labels to spare track (bit 2 set). Alternate and defective track and head fields are cleared to zero.

This directive always terminates the directive string.

a specifies the track coordinate.
b specifies the head coordinate.

5.3.14 "W" Directives

..,W,..
..,Wn,..

Inserts a Write I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The utility program writes the number of bytes specified by n, or if n is not specified, uses the value specified in parameter W as the byte count.

The Write command uses the current track, head, and sector coordinates in the sector target address register (STAR).

..,WLS,..
..,WLS,..='arg_1,arg_2,...,arg_n',..

Inserts a Write Sector Label (WLS) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,WLT,..
..,WLT='arg_1,arg_2,...,arg_n',..

Inserts a Write Track Label (WLT) I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. The default byte count is 30 bytes. If command chain (CC) is specified, IOCD word 2, bit 1, is set and the next directive is processed. For a description of the optional specifications, refer to IOCD Flag Byte Arguments and Track and Sector Flag Byte Arguments.

The legal arguments for this directive are: CC, SLI, NSIO

..,WRn,..

Inserts a Write I/O command doubleword (IOCD) in the I/O command list (IOCL) queue. If n is not specified, one sector of parameter W bytes is written to the media. If n is specified, then n sectors of parameter W bytes is written to the media.

5.3.15 "X" Directive

..,XDT='THS(a,b,c)',..

This directive allows for the deletion of entries from the Deallocation Track Table. The System Disc Flag must be OFF or the data is not written to the media with the deleted entry.

This directive always terminates the directive string.

a specifies the actual starting track coordinate and not the absolute binary address.
b specifies the actual head coordinate and not the absolute binary address.
c specifies the actual starting sector.

Note: Coordinates a, b, and c must be equal to the first coordinate shown in the Deallocation Track Table. The program searches the first occurrence of the coordinate in the Deallocated Track Table and deletes that entry from the table.

5.4 Directives Issuing Start I/O Instructions

Directives that do not have a string argument of CC (Command Chain), DC (Data Chain), or NSIO (do not perform a Start I/O) cause the utility program immediately to execute a Start I/O instruction once the directive has been processed and inserted into the IOCL queue. Directives that have string arguments of CC (Command Chain) or DC (Data Chain) insert the I/O command doubleword (IOCD) into the I/O command list (IOCL) queue and process the next directive.

5.5 Directive Arguments

Arguments are appended to specified directives. The argument syntax is as follows:

..,directive(arg_1, arg_2,...,arg_n),..

There are six categories of arguments:

- IOCD Flag Byte Arguments
- IOCD Byte Count Arguments
- Test Sector Target Address Register (STAR) Arguments
- Load Mode Register Arguments
- Track and Sector Flag Byte Arguments
- Access Control Flag Byte Arguments

Descriptions of the six argument categories follow.

5.5.1 IOCD Flag Byte Arguments

The I/O command doubleword (IOCD) Flag Byte arguments specify modifications to be made to an individual IOCD within the I/O command list (IOCL) queue. IOCD Flag Byte arguments may be specified in any order. Individual directive derivatives should be consulted to define the valid IOCD Flag Byte arguments. The IOCD Flag Byte arguments are defined as follows:

DC Data Chain

Sets IOCD word 2, bit 1, indicating that if the byte count equals zero, the utility program proceeds with the next IOCD in the IOCL queue. The next IOCD must be the same type of command as the data-chained command.

CC Command Chain

Sets IOCD word 2, bit 1, indicating that if the byte count equals zero or Channel End and Device End is detected, the utility program proceeds with the next IOCD in the IOCL queue. The next IOCD may be any valid IOCD.

SLI Suppress Incorrect Length Indication

Sets IOCD word 2, bit 2, indicating that Incorrect Length Status presentation should not be reported if detected.

SKIP Skip

Sets IOCD word 2, bit 3, indicating the number of bytes specified are to be skipped (not written to memory). This flag is valid only for read IOCDs.

5.5.2 IOCD Byte Count Arguments

BC(n) Fixed Byte Count (n)

Conditions the I/O command doubleword (IOCD) byte count (IOCD word 2, bits 16 through 31) to the value specified by n. The minimum byte count for write/read operations is four bytes.

BCPW Variable Byte Count (PW)

Conditions the I/O command doubleword (IOCD) byte count (IOCD word 2, bits 16 through 31) to the value in parameter W. The minimum byte count for write/read operations is four bytes. The byte count for sector label write/read operations is 480 bytes for a 16-sector track format and 600 bytes for a 20-sector format.

5.5.3 Test STAR Specification Arguments

TTRK(n) Test Sector Target Address Register (STAR) Track

Conditions the data address word of the Test STAR command to the track coordinate specified by (n).

THD(n) Test Sector Target Address Register (STAR) Head

Conditions the data address word of the Test STAR command to the head coordinate specified by (n).

TSEC(n) Test Sector Target Address Register (STAR) Sector

Conditions the data address word of the Test STAR command to the sector coordinate specified by (n).

Inserts the current directive into the IOCl queue and then returns to the user or program directive. This enables an IOCL queue to be built without performing a Start I/O instruction upon processing the last directive.

5.5.4 Load Mode Register Arguments

The Load Mode Register (LMR) specification arguments condition the data address byte of a Load Mode Register command with the values specified in the string. If no value is given in the parentheses, the default value is used by the argument.

Read Offset Modes

The normal setting for mode register bits 0 through 3 is zero. These bits must be zero if a disc write operation is specified. Failure to observe this restriction causes a program check, I/O program termination, and a mode check (MOCK) sense buffer indication.

The offset bits provide a normal setting (all zeros) and eight offset positions. If attempting to recover data from a removable disc pack drive, it is sometimes necessary to use the offset functions to obtain an error free disc read operation. The procedure often used is to attempt to read three or more times at the normal position, then three or more times at both plus and minus track offset positions, then three or more times at both plus and minus read time offset positions, and finally, three or more times at each of the four double offset positions. This process uses the four offset bits to select points in an eight-point rectangle surrounding the normal read position. Since some errors are likely to be intermittent and not repeatable, three or more tries should be made at each position.

CAR$\left\{\begin{matrix}(+)\\(0)\\(-)\end{matrix}\right\}$

If specified, the CAR argument conditions the mode register byte to either set or reset the carriage offset bit. The default condition is carriage offset bit 0 reset. The subfield values within parentheses follow:

(+) Sets the Carriage Offset, bit 0, and resets the positive-negative offset, bit 1.

(0) Resets the Carriage Offset, bit 0, and resets the positive-negative offset, bit 1.

(-) Sets the Carriage Offset, bit 0, and sets the positive-negative offset, bit 1.

RDT{(+) (0) (-)}

If specified, the RDT argument conditions the Mode Register byte to either set or reset the associated read timing bits. If no values are specified in parentheses, the read timing offset bit 2 is reset. The subfield values within parentheses follow:

(+) Sets the read timing offset bit 2, and resets the positive-negative read timing strobe, bit 3.

(0) Resets the read timing offset, bit 2, and resets the positive-negative read timing strobe, bit 3.

(-) Sets the read timing offset, bit 2, and sets the positive-negative read timing strobe, bit 3.

DIAG{(ON) (OFF)}

If specified, the DIAG argument conditions the Mode Register byte to either set the utility program mode bit 4 by using the value ON, or resets the utility program mode bit 4 by using the value OFF. The default condition is OFF if no subfield value is used.

MHD{(ON) (OFF)}

If specified, the MHD argument conditions the Mode Register byte to use either the moving-head disc ON or the fixed-head disc OFF for I/O operations. Bit 6 is set if the fixed-head disc is to be accessed. The default condition is moving-head disc if no subfield is used.

RES{(ON) (OFF)}

If specified, the RES argument conditions the mode register byte to either use the reserved track mode ON by setting bit 5 or not use the reserved track mode OFF by resetting bit 5.

5.5.5 Track and Sector Flag Byte Arguments

The track and sector flag byte specification arguments specify the modification to be made to a particular track and sector label. A prototype of a sector or track label is built in the output buffer with the flag bytes set to the specified arguments. The number of prototype sector labels which is built in the output buffer depends on the number of specified bytes. The following byte counts are recommended:

- 600 bytes for 20-sector/track format
- 480 bytes for 16-sector format

Flag byte consistency should be maintained. A track cannot be designated partially as an alternate track and partially as a good track.

GOOD — Good Track

Conditions flag byte 1, bit 0, in the output buffer and resets any other bits in the flag byte. Also, any data in the alternate/defective track and head field is set to zero.

ALTC(n) — Alternate Track

Conditions flag byte 1 by setting the alternate track bit 1 in the output buffer and sets the defective track parameter into the appropriate slot for either track or sector labels. The parameter within the parentheses should point to the associated defective track for which this track is being conditioned as an alternate track.

ALTH(n) — Alternate Head

Conditions flag byte 1, bit 1, in the output buffer and sets the defective head parameter into the appropriate field for either track or sector labels. The parameter within the parentheses should point to the associated defective head for which this head coordinate is being conditioned as an alternate track.

The track and head coordinates of a defective track point to the alternate track coordinates. The track and head coordinates of the alternate track must point to the defective track coordinates.

LFSC — Spare Track

Conditions flag byte 1 in the output buffer by setting the spare track, bit 2.

LFRC — Reserved Track

Conditions flag byte 1 in the output buffer by setting the reserved track bit 3.

DEFC(n) — Defective Track Cylinder

Conditions flag byte 1 by setting the defective track, bit 4, in the output buffer and sets the alternate track parameter into the appropriate field for either track or sector labels. The parameter within the parentheses should point to the alternate track.

DEFH(n) — Defective Head

Conditions flag byte 1 by setting the defective track, bit 4, in the output buffer and sets the alternate head parameter into the appropriate field for either track or sector labels. The parameter within the parentheses should point to the alternate head.

LAST{(ON) / (OFF)} — Last Track

Conditions flag byte 1 in the output buffer by setting the last track, bit 5.

FLOK{(ON) (OFF)} Write Lock

Conditions flag byte 2 in the output buffer by setting the write lock bit, 0, for either track or sector labels. Refer to Access Control Flag Byte Arguments.

FPRO{(ON) (OFF)} Write Protect

Conditions flag byte 2 in the output buffer by setting the write protect bit, 1, for sector labels only. Refer to Access Control Flag Byte Arguments.

5.5.6 Access Control Flag Byte Arguments

The Access Control Flag Byte (track label flag byte 2) arguments specify modifications to be made to a particular track and sector label. The byte contains the Write Protect bit and the Write Lock bit, the remaining six bits must be zero. The arguments are valid for the following directives only:

..,LLS='arg_1,arg_2,...,arg_n',..	Load Sector Label
..,LLT='arg_1,arg_2,...,arg_n',..	Load Track Label
..,RLS='arg_1,arg_2,...,arg_n',..	Read Sector Label
..,RLT='arg_1,arg_2,...,arg_n',..	Read Track Label
..,WLS='arg_1,arg_2,...,arg_n',..	Write Sector Label
..,WLT='arg_1,arg_2,...,arg_n',..	Write Track Label

5.6 Parameter Descriptions

The Gould RPU Disc Processor Diagnostic uses the following set of unique parameters (identified A through Z) which are manipulated by the P directive:

Parameter	Use	Description
A	Reserved	Used by the Diagnostic Executive Program (DEXP).
B	Available	Available for general use.
C	Available	Available for general use.
D	Reserved	Contains the current output buffer word 1 data pattern if the output buffer word 1 has been set using a ..,PD( )k,.. directive.
E	Reserved	Contains the number of sectors on the disc drive (sector modulus).

Parameter	Use	Description
F	Reserved	Contains the number of heads on the disc drive (head modulus). Refer to Disc Drive Test Parameters.
G	Reserved	Contains the number of cylinders on the disc drive (cylinder modulus). This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Test Parameters.
H	Reserved	Contains the current head address coordinate.
I	Reserved	Contains the current output buffer word 1 data pattern if the output buffer word 1 has been set using a ..,PI( )k,.. directive.
J	Reserved	Contains the highest valid sector address (parameter E minus 1).
K	Reserved	Contains the highest valid head address (parameter F minus 1).
L	Reserved	Contains the starting cylinder address to be used by the utility program. The starting cylinder address will default to zero. Refer to Disc Drive Test Parameters.
M	Available	Available for general use.
N	Reserved	Contains the number of bytes to use when operating with the diagnostic mode bit set in the mode register.
O	Reserved	This value is normally set to parameter G minus 5. This parameter is used as the ending test parameter by the Gould RPU Disc Processor Diagnostic.
P	Reserved	Contains the current track, head, and sector address coordinates: B0 through 15 — track address coordinate B16 through 23 — head address coordinate B24 through 31 — sector address coordinate
Q	Reserved	Contains the cylinder address of the designated diagnostic cylinder boundaries. This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Test Parameters.
R	Reserved	Contains the record count for ..,WR,.. and ..,RR,.. directives.
S	Reserved	Contains the current sector address coordinate.
T	Reserved	Contains the current track address coordinate.

Parameter	Use	Description
U	Reserved	Contains the highest valid cylinder address to be used by the utility program. This cylinder address is obtained from the ..,MODn,.. directive. Refer to Disc Drive Test Parameters.
V	Reserved	Contains the number of bytes in one sector.
W	Reserved	Contains the implicit transfer count for write and read data directives.
X	Reserved	Contains the default hold duration for the ..,H,.. and ..,HS,.. directives.
Y	Available	Available for general use.
Z	Reserved	Contains an indirect address for the ..,JZ,.. directives.

5.7 Disc Drive Test Parameters

The following table shows the value of the parameters used by the utility program. The controlling factor for the utility program is the model number, which determines the number of cylinders and heads that are verified.

Parameter	Description
F	The maximum number of heads on the drive.
G	The maximum number of cylinders on the drive.
K	The highest valid head address.
L	The starting cylinder the utility program uses.
O	This parameter normally is set to parameter G minus 5. The Deallocation Track Table normally is written to parameter G minus 4.
Q	This parameter normally is set to G minus 3 and is in the Deallocation Track Table indicating the starting address of the Diagnostic Cylinders.
U	The highest valid testable track the utility program uses.

The parameters described in this section are the default parameters determined during the execution of the Gould RPU Media Verification Program or upon completion of the ..,ZAP,.. directive in the diagnostic program.

However, the Gould RPU Media Verification Program may move the diagnostic cylinders or the ending test cylinder if the initial parameter settings for the Deallocated Track Information or the Diagnostic Tracks are detected to be faulty because of address mark errors, Disc Format Errors, or No System Interrupt are received from an I/O command.

The diagnostic cylinder parameters and end of media for the diagnostic are written to the listed output (LO) file using the ..,DBT,.. directive.

5.8 Model Number Drive Test Parameters

Drive Model n	Parameters F	G	K	L	O	Q	U
8104 (Fixed head)	4	64	3	0	59	61-62	62
8117 (Cartridge) Removable	1	823	0	0	818	820-821	822
8117 (Cartridge) Captive	1	823	0	0	818	820-821	822
8122 (Winchester)	5	823	4	0	818	820-821	822
8127 (Winchester)	10	823	9	0	818	820-821	822
8136 (Cartridge 9in)	5	823	4	820	818	820-821	822
8146 (Winchester 9in)	10	823	9	820	818	820-821	822
8155 (Winchester)	40	843	39	0	839	840-841	842
8420 (Winchester 8in)	5	823	4	0	818	820-821	822
8424 (Winchester 8in)	8	1024	7	0	1019	1021-1022	1023
8858 (Winchester 9in)	24	711	23	0	707	708-709	710
9342 (Removable)	5	823	4	0	818	820-821	822
9344 (Removable)	19	823	18	0	818	820-821	822

CHAPTER 6

I/O COMMAND TRACE

6.1 Tracing I/O Activity

The Gould RPU Media Verification Program contains optional capabilities that provide visual representation of I/O activity which augments error message reports.

If option 17 (Enables I/O Command Trace) is set, the I/O Command Trace records all I/O instructions performed to the listed output (LO) file.

6.2 I/O Command Trace

The I/O Command Trace consists of:

1. The task separation banner

```
<<<<'taskname '----hh:mm:ss>>>>
```

2. The test header

```
IOCLQP:  aaaaaaaa
IOCD (#) (1) oobbbbbb (2) ff00cccc
                    .
                    .
                    .
                  etc.
```

aaaaaaaa	specifies the real main memory address of the I/O command list (IOCL) address.
#	specifies the relative I/O command list (IOCD) decimal slot number within the IOCL Queue.
oo	specifies the order byte contents (IOCD word 1, byte 0).
bbbbbb	specifies the real main memory buffer address.
ff	specifies the flag byte contents of the (IOCD word 2, byte 0).
cc	specifies the hexadecimal byte count.

ould Inc., Computer Systems Division
901 W. Sunrise Blvd.
ort Lauderdale, FL U.S.A. 33313-4499
elephone (305) 587-2900

GOULD
Electronics